University of Alberta

CONTENT-BASED SUB-IMAGE RETRIEVAL USING RELEVANCE FEEDBACK

by

Jie Luo

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-95809-4
Our file Notre référence
ISBN: 0-612-95809-4

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

An approximate answer to the right question is worth a great deal more than a precise answer to the wrong question.

– John Tukey

To Mom and Dad,
Wherever I am, I always love you.
To Kathy and James,
Without your friendship, I would be lost.
Without your friendship, I would not have grown.

Acknowledgements

First of all, I would like to thank all those I care about back East, but especially Mum, Dad and my elder Uncle, for being supportive when I decided to abandon the eastern academic and corporate world for a graduate degree out West. I would also like to thank iCORE and the Department of Computing Science at the University of Alberta for making the life of a graduate student so attractive with their funding.

This research would not have been possible if it were not for the constant support and encouragement of my remarkable supervisor, Dr. Mario A. Nascimento. He gave me the chance to further my research on information retrieval and trusted that I could accomplish something of merit, although initially I had some adaptation problems to a different cultural and academic environment. He gave me as much edification on research methods and freedom as I needed to get my research work done. I am also grateful to the University of Alberta Database Research Group for providing me with a forum to present my ideas and receive feedback. Among the members of the group, I would like to especially thank Yue Zhang for the technical advice early on, and Meng Xue's contribution in early stage of this research. Besides, I acknowledge the reviewers of ACM-MMDB 2003 and CIVR 2004 for their insightful comments about my publications.

My thanks to my schoolmates in the Department of Computing Science at the University of Alberta – Lihang Ying, Yongjie Liu, Fan Deng, and Jia Li, for their helpful technical and daily life discussions. Finally, many thanks also go to the dear friends I met in the States – Xin Liu, Mingjie Yang, and here in Edmonton – Xiaomeng Yang, Yiqian Wang, Carrissa Halton, Siauw Ng, and James Scott, for sharing my happiness, sadness, and worries, as well as being supportive, comfortable, and always close to my heart.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Alberta Informatics Circle of Research Excellence (iCORE), and the Canadian Cultural Content Management Research Network, a Network advanced through Heritage Canada's New Media Research Networks Fund.

Table of Contents

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Challenges . . . 3
- 1.3 Contributions . . . 5
- 1.4 Organization . . . 5

2 Related Work for Content-Based Sub-Image Retrieval 7
- 2.1 Color-Based Image Retrieval . . . 8
  - 2.1.1 Color Spaces . . . 8
  - 2.1.2 Color-Based Image Description and Representation . . . 9
  - 2.1.3 Distance Function . . . 11
  - 2.1.4 Similarity Search . . . 12
- 2.2 Region-Based Image Retrieval . . . 14
  - 2.2.1 Techniques for RBIR . . . 14
  - 2.2.2 RBIR vs CBsIR . . . 16
- 2.3 Recent Work in Content-Based Sub-Image Retrieval . . . 17
  - 2.3.1 Partition-Based CBsIR . . . 17
  - 2.3.2 Point-Based CBsIR . . . 19

3 CBsIR Framework via Hierarchical Tree Matching 21
- 3.1 HTM's Hierarchical Partition and Tree Structure . . . 22
- 3.2 Feature Extraction using Average Color and Vectorial Distance . . . 24
  - 3.2.1 Average Color . . . 24
  - 3.2.2 Vectorial Distance Functions . . . 24
- 3.3 Feature Extraction and Distance Measure by Border/Interior Pixel Classification . . . 27
  - 3.3.1 Image Description . . . 28
  - 3.3.2 dLog Distance Function . . . 30
- 3.4 Feature Extraction and Vectorial Distances for Sebe et al's method (IHD) . . . 34
  - 3.4.1 Color Indexing . . . 34
  - 3.4.2 Inter Hierarchical Distance (IHD) . . . 35
- 3.5 Efficiency and Storage Consideration . . . 36
  - 3.5.1 Feature Representation . . . 36

3.5.2 Index Sequence 38
3.6 Search Strategy 39
3.6.1 Search by Hierarchical Tree Matching 39
3.7 Performance Study 42
3.7.1 Performance Measures 42
3.7.2 Experimental Setup 44
3.7.3 Results Analysis 46
3.8 Summary 54

4 Supervised Learning in Content-Based Sub-Image Retrieval 56
4.1 Relevance Feedback in Other Image Retrieval Scenarios 57
4.1.1 Learning in traditional CBIR 57
4.1.2 Learning in RBIR 59
4.2 Relevance Feedback for CBsIR 60
4.2.1 Tile Re-Weighting Scheme 61
4.2.2 Query Feature Update 64
4.2.3 Image Similarity 66
4.3 Performance Measures 67
4.4 Experiments and Results 69
4.5 Summary 77

5 Conclusions and Future Work 79

A Simulation of HTM's Search Process 82

Bibliography 88

List of Tables

3.1 Average actual rank and group rank. 44
3.2 Six sample images with query sub-images indicated by the white frames. 45
3.3 Performance of different methods using average actual rank. 47
3.4 Performance of different methods using group rank. 47
3.5 Comparison of the distributions of relevant images using average actual rank. 48
3.6 Search results for six sample query sub-images by HTM. (Note that the size of query sub-images and retrieved images is changed for viewing purpose.) 50
3.7 Comparing query processing efficiency for HTM and IHD. 51
3.8 Retrieving original images using IHD and different feature representations with HTM. 52
3.9 Comparison of the distributions of original images using different feature representations. 54
3.10 Comparing query processing efficiency using different BIC histograms with the HTM scheme. 54

4.1 Cumulative/New/Actual Recall and Precision 69
4.2 Comparison of retrieving the original images using BIC by feedback iterations. 70

List of Figures

2.1 Color spaces 9

3.1 Hierarchical partition of an image with the resulting tree structure and one possible corresponding index sequence for storage. 23
3.2 Simple example of tree structures for database image and query sub-image. 27
3.3 GCH vs. BIC Histograms for texture discrimination. 31
3.4 Two examples of the result by the BIC pixel classification. 31
3.5 Index sequence. 38
3.6 Index sequence without redundant tile at the lowest level of the hierarchical structure. 38
3.7 Search algorithm of the HTM method. 40
3.8 Determining the best matching image (sub)tree. 41
3.9 Sample query sub-images and their original images (answers) with average actual ranks using different methods. (Lower ranks are better.) 49
3.10 Sample query sub-images and answers with ranks using the IHD method and different image indexing methods with HTM. (Lower ranks are better.) 53

4.1 Initial set of retrieved images with user's feedback. 63
4.2 Comparison of tile penalty for database image I_1 before and after feedback. 65
4.3 A sample query (sub)image and its relevant answer set. 70
4.4 Effectiveness measures by actual recall, cumulative recall and new recall using 64 quantized colors in BIC. 71
4.5 Effectiveness measures by actual precision, cumulative precision and new precision using 64 quantized colors in BIC. 72
4.6 Results of online demo using 64 quantized colors in BIC for a sample query after the first iteration. (The user has given feedback once.) 73
4.7 Results of online demo using 64 quantized colors in BIC for a sample query after the second iteration. 74
4.8 Effectiveness measures by actual recall, cumulative recall and new recall using 16 quantized colors in BIC. 75
4.9 Effectiveness measures by actual precision, cumulative precision and new precision using 16 quantized colors in BIC. 76

4.10 New recall (defined from the second iteration) comparison using 64 quantized colors and 16 quantized colors in BIC. 76
4.11 New precision (defined from the second iteration) comparison using 64 quantized colors and 16 quantized colors in BIC. 77
4.12 Comparing query processing efficiency using different BIC histograms at each iteration. 78

A.1 Tree structures modeling the hierarchical partition of the image. . . 83
A.2 Matching the full trees of the query and the database image. 84
A.3 Matching the sub-trees of the query and a certain tile at the second level of the hierarchical partition. 85
A.4 Obtained distances between the query and each tile of the database image after two kinds of match. 86
A.5 Image similarity measure - finding the minimum among the obtained distances as the distance between the query and the database image. 87

List of Abbreviations

BIC Border/Interior pixel Classification
CBIR Content-Based Image Retrieval
CBsIR Content-Based sub-Image Retrieval
CCV Color Coherence Vector
EMD Earth-Mover's Distance
GCH Global Color Histogram
HTM Hierarchical Tree Matching
IHD Inter Hierarchical Distance
LCH Local Color Histogram
MAM Metric Access Method
RBIR Region-Based Image Retrieval
RF Relevance Feedback
SAM Spatial Access Method
SVM Support Vector Machine
TP Tile Penalty

Chapter 1

Introduction

1.1 Motivation

Concrete visual means like images has always been preferred by human to express ideas and convey information since remote antiquity. In the current information explosion age, our reliance on visual modes of communication has further been reinforced by the recent rapid technological evolution in handling digital data. This can be witnessed in the overwhelmingly growing amount of digital image data with the development of the world wide web. Thus, image databases are becoming more and more common in several diverse application domains, such as multimedia search engines, digital libraries, medical and geographical databases, etc. Although constructing very large image databases has become fairly easy with the advances of techniques for acquisition, transmission and storage of images, the information stored there is virtually useless if not organized. In this scenario, searching for a certain image from a large image repository is just like looking for a book from a huge library without the aid of catalogs. All these factors have stimulated great interest in image retrieval techniques.

But how difficult is the problem of searching and retrieving images? Unfortunately, traditional text retrieval methods are not suitable for images because of the dimensionality difference between images and text as well as the data size difference between them (image data is much larger than text). Moreover, it should emphasize the fact that in some sense words themselves are semantic "objects", while the image data needed to be processed and interpreted to extract the percep-

tual meaning, which cannot be achieved by textual indexing techniques [14].

Early image retrieval [6] was performed based on short descriptions as a set of content-independent attributes (file name, format, category, size, author's name, and disk location) of the images. However, this approach limits the queries to those based on existing attributes. Another alternative is to use manual text annotations or keywords so that classical information retrieval (IR) techniques can be used to search images indirectly. But this approach still has problems like ambiguity, incompleteness and subjectiveness. Since image data is very rich in information, to capture the content of an image using just a few keywords is not feasible, not to mention the tedious work involved in the annotation process.

A more effective and automatic approach is the so-called *content-based image retrieval* – CBIR, which consists of using low-level image features to represent, compare and retrieve images. Most CBIR systems [15] follow the two-step approach to search image databases [8]. Firstly (*indexing*), a feature vector representing certain essential properties of the image is extracted and stored as metadata for each database image. Secondly (*searching*), given a query image, images most similar to the query image are returned to the user by comparing the feature vectors of database images with that of the query image. These CBIR systems all belong to the *Query-By-Example* (QBE) paradigm.

While most CBIR systems retrieve images based on a full image comparison, i.e., given a query image the system returns overall similar images. However, users can also be interested in *object searching* [23]. In this case, the user provides a sub-image query (perhaps an object) and the system should retrieve images that *contain* the query (according to human perception) from the image database. The sub-image query can be also an image itself. This task, which we call *content-based sub-image retrieval* (CBsIR), is difficult to cope with by a variety of effects (such as size variation and different viewing positions, etc.) that cause the target sub-image to have dramatically different appearances in different images. A problem associated to the CBsIR task is to how to locate the sub-image inside a database image effectively and efficiently.

Besides the basic CBsIR tasks, several related problems also need to be ad-

dressed. Most CBIR and CBsIR systems automatically generate low-level image features such as color, texture, shape, etc, for image indexing and retrieval, which do not capture the semantics of images. And there is no effective method yet to automatically generate good semantic features of general images. When the system retrieves some images that are irrelevant to the sub-image query according to the user's judgement, the user might want to provide feedback information about the relevance of the obtained results to reinforce the accuracy of future retrievals. Then the CBsIR system should process the feedback information efficiently and return better result by the user's intention.

1.2 Challenges

A large number of challenges exist in the CBIR research domain. This thesis deals with the challenges described in the following two paragraphs.

The feature extraction of an image database is to compute a n-dimensional vector for a feature based on some image analysis. Color, texture, shape and spatial information are the most commonly used *low-level features* in image retrieval systems. The n components of a feature vector may be derived from one visual features or a combination of several ones, e.g., [22]. A good low-level feature for an image should be able to preserve the perceptual similarity, fast to compute and small in size. The feature vector not only affects the retrieval efficiency, but also affects the design of indexing data structures when the size of the image database becomes very large, e.g., the huge image collection available from the Internet, which makes the CBsIR task especially challenging. The perceptual similarity determines the effectiveness of the feature for retrieval purpose. Being a contrast to low-level image features, semantic categories depicted in images are called *high-level concepts*. However, a big gap exists between low-level image features and semantic contents of images. In addition, human perception is subjective and task-dependent. All these limit the retrieval accuracy of most CBIR (and CBsIR) systems. While high-level concepts could help facilitate the human-computer interaction, they are almost impossible to extract without human assistance.

A *distance function* measures the *similarity* between two given images by computing the difference of the two corresponding feature vectors. The greater the distance, the smaller the similarity. The distance function is usually defined as City-Block (L_1) norm, Euclidean norm (L_2), or weighted Euclidean norm [10]. Vectorial distances are efficient in comparing histograms and allow the use of spatial or metric access methods to speedup query processing [53]. However, they have also well-known limitations. One of such limitations is that a high value in a single histogram bin dominates the distance between two histograms, no matter the relative importance of this single value [36][50]. This kind of limitation causes distortion in retrieval results when comparing images having a large background with the same color but a different foreground with images having the same foreground (a high degree of semantic similarity) but a large background with a different color.

Another question to consider is, how precisely should we measure the distance between database images and the sub-image query? If we choose a high precision, it surely distinguishs distances in a finer granularity. However, it should also be noticed that there are so many approximations in the whole retrieval process and humans often do not have such a fine distinction between perceived similarities. Thus, a lower precision might be better suited. This yet results in the increase in the number of tied distances. Different degrees of distance precision yields different distance figures. The ranking based on these distances often become confusing and could handicap the correct understanding of the investigated methods.

In addition, once it is determined that an image contains a sub-image query, it becomes necessary in many cases (like tracking objects in videos) to find the place of the sub-image inside the database image. Note that there should be no restriction as to where the sub-image query may be within a relevant image in the CBsIR system. Because of the lack of accurate and efficient image segmentation process for large, arbitrary and heterogeneous image databases, the sub-image queries may have to be located in unsegmented images. The problem of how to locate the sub-image effectively and efficiently is thus made more difficult.

1.3 Contributions

In this work, we investigate the problem of content-based sub-image retrieval (CBsIR) to find database images that contain the query sub-images in two ways.

First, we propose a new method called Hierarchical Tree Matching (HTM) for the CBsIR problem. The highlights of this approach are: (i) it uses a fixed decomposition without relying on image segmentation (typically not an accurate process), (ii) the hierarchical partition encodes the local spatial information as well as global distribution of colors in the image, (iii) the multi-scale representation is small in size and stored in the format of an index sequence (allowing fast access during the search phase), (iv) a search strategy is designed to achieve effective and efficient retrieval based on the multi-scale representation. Experimental evidence, tested on an image database of over 10,000 images, shows the new approach outperforms other related CBsIR approaches and achieves a good balance of accuracy and efficiency.

To further improve retrieval results, we also address how relevance feedback can be used to enhance the performance of HTM-based CBsIR system. We present a tile re-weighting scheme that assigns penalties to each tile consisting of database images and updates those of all relevant images using both the relevant (positive) and irrelevant (negative) images identified by the user. Learning is effected by modifying the query vector to incorporate the positive examples based on the update of their tile penalties during the feedback iteration. Besides, a new similarity distance between an image and the sub-image query is also learned by using a weighted metric by the tile penalty, which is possible to shorten the distance between the query and relevant images and elongate the distance between the query and irrelevant images. Our results suggest that this learning method is quite effective for the CBsIR system.

1.4 Organization

Chapter 2 discusses related research literatures in (color-based) image retrieval, distinguishing the problem of content-based sub-image retrieval (CBsIR) from other active research domains in content-base image retrieval (CBIR) and giving a brief

survey of related CBsIR systems. In Chapter 3, we propose the Hierarchical Tree Matching approach for CBsIR. We investigate the new method by discussing in detail each step in the whole retrieval process. Chapter 4 describes how to use relevance feedback combined with the hierarchical tree matching scheme to improve the results of content-based sub-image retrieval. Corresponding experimental results are shown in the above two chapters respectively. Finally, Chapter 5 concludes the thesis and offers direction for future work.

Chapter 2

Related Work for Content-Based Sub-Image Retrieval

Thus far, a large amount of research published within the area of content-based image retrieval (CBIR) [32][53] deals with full-image retrieval, where typically one provides a query image and the CBIR system finds the most similar images from an image database. The notion of similarity is usually such that the returned images should resemble the query in an "overall"manner. An also interesting, though so far much less explored, problem is that of finding images that *contain* the query images, i.e., images where the query image is part of the overall image. We term this problem Content-Based Sub-Image Retrieval (CBsIR); and define it as follows [39]: given a sub-image query Q and an image database S, retrieve from S those images Q' which *contain* Q according to some notion of similarity. It is important to clarify that the sub-image retrieval problem is a distinct branch of the image retrieval domain, which has its own characteristics and merits in various applications.

In this chapter, we discuss existing techniques for two branches in CBIR from different points of view. The use of low-level features, color in particular, is useful for large and heterogeneous collections of images, where images belong to several distinct, non-related semantic and visual domains. Since color is also used as image feature in our CBsIR system, a survey of existing color-based image retrieval approaches providing an overview of the background for our CBsIR system is presented in Section 2.1. Although our approach for the CBsIR problem decomposes the images into tiles and define distances between feature vectors ex-

tracted from different image tiles, this is not the same as region-based image retrieval (RBIR) investigated elsewhere, e.g., [34][41]. Nonetheless, some region-based image retrieval methods are also reviewed in Section 2.2 for completeness. Moreover, within the context of CBsIR, methods proposed in related literatures are investigated in Section 2.3 and further compared with our CBsIR approach by experiments in Chapter 3.

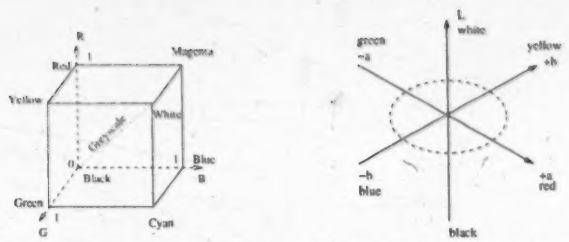
2.1 Color-Based Image Retrieval

The choice of the right image features for an image retrieval system is important since image features affect every aspect of the whole retrieval process. Most of the CBIR systems explore low-level image features like color, texture, shape, etc., since they can be extracted automatically. Color is the most commonly used low-level feature, possibly because color is immediately perceived by humans and related concepts are easy to understand and implement. Besides, color is one of the most prominent perceptual features in a large majority of image domains and using color information can often achieve satisfactory results. Most commercial CBIR systems include color as one of their image features (e.g., QBIC of IBM [9], Virage[11], etc). This section is thus mostly concerned with color-based image retrieval approaches.

2.1.1 Color Spaces

The color of a pixel in a digital image is typically represented by three values, one for each channel of the chosen color space. A color space is a specification of a 3D coordinate system and a subspace within that system where each color is represented by a single point [7]. The first step in any color-based image retrieval system is to choose a color space where images will be represented and compared.

The most well-known and used color space is the RGB (Red, Green, Blue) model [7][36]. The RGB color space is device-dependent such that the displayed color depends not only on the RGB values, but also on the device specifications.



(a) The RGB color space (b) The CIE Lab color space

Figure 2.1: Color spaces

The main drawback of this model is that it is not perceptually uniform, in the sense that the differences between RGB colors do not reflect the differences perceived by human. The RGB color space is a cube shown in Figure 2.1(a), where the main diagonal represents the gray values from black to white, and any point (color) inside the cube is represented by a weighted sum of red, green, and blue [53].

Another kind of color spaces is uniform color spaces, where the numerical differences among colors are consistent with the differences perceived by human. The CIE Lab model is one such example. As shown in Figure 2.1(b), the CIE Lab color space represents the differences of three elementary pairs: red-green, yellow-blue and black-white. Different from the RGB color space, the CIE Lab color space is device independent.

The third kind is called the user-oriented color spaces [32][36], which are based on human perception of colors like hue, saturation and intensity. Some example of this kind are the HSI and HSV color spaces, where are device dependent.

2.1.2 Color-Based Image Description and Representation

To achieve effectiveness and efficiency in image retrieval systems, a compact and accurate description of the color distribution and the spatial distribution of colors in the digital images is needed. These descriptors can be further reduced in size by static or dynamic reduction methods.

Static methods use a fixed scheme for each image. The simplest scheme to reduce the number of colors in an images is to use a uniform and coarse quantization of each color channel. Thus the obtained colors need not be represented explic-

itly and the comparison of images is easier. However, it is possible that the colors present in an image are not uniformly distributed in the color space. It is also not appropriate for non-uniform color spaces like HSV, since similar colors may be separated and non-similar colors classified together. Another problem is that it is difficult to obtain an adequate compromise about the granularity of the quantization for the not necessarily uniformly distributed colors in the color space. Dynamic reduction methods exploit the visual content of the images and rely on image segmentation techniques to reduce both the number of colors and the number of spatial locations in an image. A typical image segmentation technique groups neighboring pixels with similar colors together into regions whose colors are the average color of their pixels. The resulted regions are more compact and meaningful since they bear high degree of color similarity and well-defined spatial location, size and shape. A sample of the image segmentation techniques used by these methods includes: boundary detection, region growing, region splitting and merging [7].

Once the description of the image is chosen, a representation of this information is the next step in image retrieval systems. Color histogram has been widely used to describe the color information of the image since it is easy to compute, relative insensitive to position and orientation changes, feasible in terms of memory usage, efficient to compare using vectorial distance functions and sufficiently accurate for retrieving images based on overall color impression. The stored information about the visual content of an image can be represented in three possible ways described next.

Global representations describe the color distribution of the whole image, ignoring the spatial distribution of colors. The most commonly used global representation is the Global Color Histogram (GCH) [32][36]. A GCH is computed by counting the number of pixels in the image having each of the quantized colors. Usually, the pixel count is normalized to avoid scaling bias [53]. However, since global color histogram ignore spatial or topological information, it has limited image discriminative power. Another alternative is to use partition-based representations to describe the color distribution of each partition of an image individually. Generally, the image is statically partitioned into a set of rectangular units according to some

scheme, and a Local Color Histogram (LCH) is used to describe each partition unit individually. In this kind of representation, extra information about spatial properties of the partition units such as size, shape and spatial location need not to be saved, since it is easy to obtain that from the predefined scheme. Some partition-based approaches also use other kinds of color histograms [28][58] to introduce some spatial information about the visual content of the images decomposing them into spatial cells according to a fixed scheme. Apart from the above two representations, regional representation exists for object-level image retrieval, which will be discussed in more detail in section 2.2.

Based on the image representation, existing color-based image retrieval techniques can be classified into three main categories: (1) global approaches (e.g. [32][36]), (2) partition-based approaches (e.g. [28][58]), (3) region-based approaches (e.g. [34][41]). Each of these categories poses a distinct compromise among the complexity of the image analysis algorithm, the amount of space required to represent the visual features extracted from images, the complexity of the distance function used to compare these features, and the retrieval effectiveness.

2.1.3 Distance Function

The success of the image retrieval problem depends mainly on two factors. One is the stability of image features used, the other one is the characteristics of the distance function used for comparing the image features. The distance function affects directly the query processing time and the retrieval accuracy. The better the distance simulates the human perception of similarity, the more effective is the image retrieval system in retrieving images related to the user's need. The computational complexity of the distance function is also considered an important factor when processing a visual query. Moreover, the distance function restricts use of different filtering techniques and/or access methods can be used to speedup query processing.

Some well-known vectorial distance functions include [53]:

$$L_1(City - Block) : L_1(a, b) = \sum_{i=1}^{k} |a_i - b_i|$$

$$L_2(Euclidean) : L_2(a,b) = (\sum_{i=1}^{k} |a_i - b_i|^2)^{1/2}$$

$$L_\infty(Chebyshev) : L_\infty(a,b) = max_{i=1}^{k} |a_i - b_i|$$

where $a = (a_1, a_2, \ldots, a_k)$ and $b = (b_1, b_2, \ldots, b_k)$, both are k-dimensional feature vectors.

Modeling visual features in a vectorial space has the advantage that the geometric distance used to compare two vectors are computationally simple. However, there are other cases, such as in region-based image retrieval systems, where it is not possible to model complex image retrieval systems in a vectorial space. In such scenario, a metric space, where there is no restriction about the representation of the visual features, is used instead. A recent metric proposed to measure the distance between two distributions of some random variables in an image, such as color histograms, is the Earth-Mover's Distance (EMD) [31]. EMD reflects the minimal amount of work that must be performed to transform one distribution into the other by moving the "distribution mass" around. It comes from the transportation problem in combinatorial optimization. EMD can be computed by solving a linear programming problem, thus it is computationally expensive. In addition, recent research in psychology and computer vision implies that human perception of similarity contradict in different ways with the metric axioms, which are believed to be too restrictive in the context of similarity search. One of the most criticized metric axiom is the triangular inequality, coincidentally the most important axiom for indexing purposes [30] but difficult to enforce in complex matching algorithms that are statistically robust. This raises serious questions about the extent to which existing work on classification can be applied using complex models of similarity. Thus, as a possible solution, non-metric distances turn up in many application domains, such as string (DNA) matching and retrieval from image databases. Some non-metric similarity measures are suggested for image classification in [27].

2.1.4 Similarity Search

Searching for target digital images differs from the usual database search. The simplest way is sequential scanning. Each image is compared against the query

image for the candidate matches to the query image. Although simple this approach does not scale since the query processing time is proportional to the database size.

In order to reduce the complexity of the searching process, filtering techniques and access methods can be used. On one hand, filtering techniques try to reduce the complete database into a much smaller subset that has to be compared using a complex function, by relying on a simpler distance that lower-bounds the original complex distance to quickly filter out irrelevant images. One of the most common reductions for filtering techniques consist in mapping a general metric space into a vector space in such a way that each element of the metric space will be represented as a point in the target vector space [48]. If the vectorial distance is a lower-bound for the original distance, then it is guaranteed that the filtering process will not filter out relevant images. An example of the reduction discussed above is the use of the average color as a filter for color histograms. Since the comparison of average colors is much more efficient than the comparison of color histograms, it is possible to quickly eliminate the majority of the irrelevant images using this simple filter.

On the other hand, access methods aim to divide the search space into several subspaces so that only a few of these subspaces need to be searched when processing a query. This may be based on using more sophisticated combinations of techniques and data structures to quickly locate the features that are relevant to a visual query. Spatial access methods (SAMs) use spatial coordinates to group and classify points in the space. These methods are very sensitive to the number of dimensions of the vectorial space. A survey on SAMs can be found in [24]. SAMs uses the absolute spatial location of objects to partition and search the vectorial space. But in a general metric space, the unique information available is the relative distance among objects. In this case, metric access methods (MAMs) [35] aim to partition the data space in regions by choosing representative elements and clustering the other elements around them. These MAMs can be classified in main categories [35] as those based on discrete distance functions and those that deal with continuous distances or as static and dynamic according to their support for insertion/deletion after the creation of the index.

2.2 Region-Based Image Retrieval

2.2.1 Techniques for RBIR

Because low-level image features have weak connections to semantic content of images, object (region)-level image retrieval has been used in an attempt that obtained regions correspond to higher-level concepts, e.g., objects, that can be easily distinguished by the user. To achieve object-level querying, most region-based image retrieval systems are based on segmentation techniques to decompose images according to their visual content. The segmentation of images yields regions with different size, spatial location and shape, being more flexible than the fixed scheme adopted in partition-based approaches. However, the comparison of segmented images is a very difficult problem because of inaccurate segmentation [41]. In general, the result of the image analysis algorithm in region-based approaches can not be used directly to represent and to compare images, since the number of segmented regions is usually very high. Because a precise description and comparison of a large number of regions are too expensive in computational terms, the image analysis result is post-processed so as to reduce the number of segmented regions and to simplify the description of the left regions. Unfortunately, this simplification certainly affects the effectiveness of these approaches. Except for some narrowly defined problem domains where domain knowledge and the *apriori* object models are available, accurate and complete segmentation on generic real world scene can seldom be achieved [25]. Some common characteristics in natural scenery, such as shade, highlight, and sharp contrast, are major challenges to image segmentation. The most common approach in RBIR systems is to compare the regions individually, e.g., [34]. In order to reduce the effect of inaccurate segmentation, recent systems like SIMPLIcity [41] and CBC [47] try to compare images using the properties of all segmented images, not only on a region-by-region basis. In the following we describe some existing work in region-based methods.

The QBIC system [9] uses a clustering process where two clusters are merged if their mutual rank falls bellow a threshold. The Euclidean distance between two clusters' mean colors is treated as the distance between clusters. A bounding rect-

angle is calculated for each connected component identified after the clustering process. Then the bounding rectangles for a given color are successively clustered into groups of rectangles that are geometrically close to each other until one rectangle remains. The distance between two regions is computed as a weighted sum of the distance between the clustered colors and the distance between the resulting hierarchical tree associated to them. Finally, the similarity between two images is measured by the average of the distances between each region of one image and its closest region in another image.

In [17], a boundary detection procedure that explores edge flow of both color and texture is used to segment images into homogeneous regions. Each region has several features, such as color, texture and shape, which are indexed separately. A query might consists of more than one of these features. And the results from individual features are sorted by a weighted similarity measure. The system deals with images of different categories by tuning a set of system parameters.

The Blobworld system [34] clusters pixels in eight-dimensional space of joint color, texture and position, which is modeled as a mixture of Gaussians. A 500 bins local color histogram in L*a*b* color space is used to represent the color distribution of each region. The images are compared based on individual regions using the weighted Euclidean distance. The retrieval task is that of finding database images that have a region similar to a given region, possibly an object, in a query image. Although it allows querying based on a limited number of regions, the query is performed by merging single-region query results.

In the SIMPLIcity project [41], images are segmented based on color and frequency features by the *k-means* clustering algorithm to group the feature vectors into classes, which correspond to regions. The IRM (Integrated Region Matching) similarity measure is used to compare images based on the properties of all segmented regions. First, a certain region of an image is allowed to match several regions of another image during the match process. After regions are matched, a weighted sum of the similarity between region pairs is computed for the image similarity measure, with weights by a significance matrix.

The CBC approach [47] applies a fully automatic clustering algorithm. Its time

complexity is in $O(nlogn)$, where n is the number of pixels in the input image. The average L*a*b* color and the spatial coordinates of the geometric center of each resulting region are extracted as image features. A distance function similar to the IRM measure as in SIMPLIcity system is adopted to compare two segmented images.

2.2.2 RBIR vs CBsIR

The sub-image retrieval problem we consider is similar to the region-based image retrieval (RBIR) discussed before, since the goal can also be to retrieve images at object-level. However, the difference between these two problems stands out as the CBsIR problem is to search for an image, given as a whole, which is contained within another image, whereas in RBIR one is searching for a region, possibly the result of some image segmentation. The former is more intuitive since users can provide a query image as in traditional CBIR, and unlike the latter, it does not rely on any type of segmentation preprocessing. Unfortunately, automatic image segmentation algorithms usually lead to inaccurate segmentation of the image when trying to achieve homogeneous visual properties. Sometimes the obtained regions are only parts of a real object which a user would likely identify by looking at the image and should be combined with some neighbor regions so as to represent a meaningful object. Thus, complex distance functions are generally used to compare poorly segmented images at query time. Also, the number and size of regions per image are variable and a precise representation of the obtained regions may be storage-wise expensive. What's more, since region-based queries are usually performed after the image segmentation and region description steps, it clearly puts some restriction on the user's expression of his/her information need depending on how good the segmentation results match the semantics of images, although the user can explicitly select any detected region as query region. In those image retrieval systems whose images are heterogeneous, rich in texture, very irregular and variable in contents, accurate regions are hard to obtain, making RBIR likely perform poorly. Whereas the seemingly simpler CBsIR with fixed partition could be a solution in such cases.

2.3 Recent Work in Content-Based Sub-Image Retrieval

In [55], T. Wang et al intend to find an effective way to perform CBsIR and ranking. Two kinds of image feature vectors: the *global color histogram* and the *autocorrelogram* [25] with L_1 and D_1 distance measures [16] are tested in the sub-image retrieval system. Another distance measure called S_1 which aims to emphasize the contribution of colors that have very different distributions between the images is proposed as well. Preliminary experiments with several distance measures for both feature vectors find that the combination of autocorrelogram feature vector and the so-called S_1 distance measure outperforms other combinations and yields excellent results for sub-image retrieval with an acceptable processing overhead. Yet more work is still needed to further understand how to achieve the CBsIR task efficiently and how the corresponding CBsIR system works. As we have pointed out in Section 2.2.2, the CBsIR systems we are concerned with do not belong in the region-based image retrieval domain, but use other categories of image retrieval approaches classified by the way how the information in an image is represented. These methods include partition-based approaches as in [28][39][43][57] and point-based approaches as in [45].

2.3.1 Partition-Based CBsIR

Image partitioning is an important factor to determine the functionality and the efficiency of image retrieval systems [38]. By breaking images into smaller and more manageable units, it usually becomes easier to compress, store, access and retrieve the image data. The partition-based approaches usually adopt a hierarchical representation of the spatial decomposition using a simple fixed strategy based on a grid of rectangular cells superimposed over the images [28][39]. The cells at distinct hierarchical levels have various sizes and overlap,which makes it possible to detect that two images whose objects are in different positions are deemed similar. Two images are compared initially at the top of the hierarchy and refined in subsequent levels. For efficiency and effectiveness, the partition-based approaches generally

stand in between two other kinds of solution to CBIR known as global approaches, which sacrifice retrieval effectiveness with the absence of spatial and topological information for high efficiency in terms of visual feature extraction, space overhead, and image comparison, and the region-based approaches using complex image processing techniques to decompose images into regions of high similarity, implying complex image analysis algorithms for feature extraction, complex distance functions for image comparison and high space overhead but an improved retrieval effectiveness. The space overhead for the partition-based approaches might be large in such cases when using the hierarchical representation of the partition structure as mentioned before. In the following, we briefly review some recent work in partition-based methods. The paper by Leung and Ng [28] investigates the idea of using the Padding and Reduction algorithms to support sub-image queries of arbitrary size based on local color information. The algorithms either enlarge the query sub-image to match the size of an image block obtained by the multi-resolution representation of the database images, or conversely contract the image blocks of the database images so that they become as small as the query sub-image. The paper presents an analytical cost model and focuses on avoiding I/O overhead during query processing time. To find a good strategy to search multiple resolutions, four techniques are investigated: the branch-and-bound algorithm, Pure Vertical (PV), Pure Horizontal (PH) and Horizontal-and-Vertical (HV). The HV strategy is argued to be the best considering efficiency. However, the authors do not report clear conclusions regarding the effectiveness of their approach (e.g., Precision and/or Recall figures).

In [39], the global feature extraction is considered to capture the spatial information within the image "regions" which are not the same concept of regions in region-based image retrieval. The average color and the covariance matrix of the color channels in L*a*b* color space are used to represent the color distribution. They apply a three-level non-recursive fixed hierarchical partition with overlapping rectangle "regions" to achieve the multi-scale representation of database images. Aiming at reducing the index size of these global features, a compact abstraction for the global features of a "region" is introduced. A new distance measure on

such abstraction is thus proposed for efficiently searching through the tiles from the multi-scale partition strategy. This distance is called *inter hierarchical distance* (IHD) since it is taken between feature vectors of different hierarchical levels of the image partition. The IHD index is a two dimensional vector which consumes small storage space. And the search strategy is a simple linear scan of the index file, which assesses the similarity between the query image and a particular database image as well as all its "sub-regions" using their IHD vectors. Finally, the minimum distance found is used to rank this database image. This approach is argued to be efficient and effective. We will compare our proposed approach with this one in Chapter 3.

The application of CBsIR to the domain of high resolution art images has been studied in [43]. The proposed approach is called the Multi-scale Color Coherence Vector (M-CCV) method, based on the use of color coherence vectors [12] extracted from image patches for the query and target images at a range of scales with multiple vector matching to find the best sub-image matches. The query sub-image may be a poor quality reproduction of part of the original and may be digitized under significantly different conditions. Tested on a collection of art images, many of which at very high resolution, the technique is demonstrated to perform well.

2.3.2 Point-Based CBsIR

Image retrieval systems of the "query by example" style usually concern the entire image. In the context of part/object-level user interest, global image descriptors are of less use. In this case, the approaches based on grey points of interest [19] and color points of interest [45] have been developed for object/sub-image retrieval tasks, which require more local descriptors, and are discussed next.

Points of interest are points extracted and characterized from color signal at once [45]. They are pixels that capture significant local features of an image, and usually locate around corners and edges of images. A local image descriptor based on color points of interest, was proposed in [45] which focuses on object or sub-image retrieval. Compared with region-based approaches in which the quality of the segmentation step is sensitive to image geometrical contents, the points of interest extraction performs well whatever the image content is. Besides, points are more

robust to geometric transformations of the image like view point changes, since the description is computed locally, and robust to partial occultation. Moreover, content-based image retrieval techniques exploit photometric information contained in the images, which just matches the definition of a point of interest - being located where this photometric information is most significant. Therefore, there exists great expectation of using points to achieve a rich and compact image characterization.

When applied to image retrieval, image matching based on points of interest needs points with good *repeatability*. The ideal interest points, which indicate local features, should be invariant to illumination change and geometrical transformation. Many point extractors exist in the literature of Computer Vision. It has been demonstrated that the Harris color detector [29] fits better for the required repeatability. The first step of the image feature indexing is to extract points of interest from the whole images by this detector. Second, the points of interest are described using photometric quantities implying color differential invariants. The resulting image characterization is argued more compact than other existing ones, since it contains more photometric information while having comparable storage cost. This characterization is also claimed to perform well for object or sub-image description, as it implies a local description of the image that is robust to image transformations. The search strategy applied in [45] consists of a voting algorithm. The vote computed for each image of the database is the function of the distances between the query points and the candidate points of the involved image. Experimental results show the success of this approach for partial retrieval on sub-images and on 3D objects as well as object retrieval under difficult conditions like viewpoint changes and occultations.

Chapter 3

CBsIR Framework via Hierarchical Tree Matching

There are two main factors that cause the limited retrieval accuracy in CBIR in general and CBsIR in particular. One is the gap between low-level image features and semantic contents of images. We will discuss how to reduce this gap in Section 4 using machine learning techniques. The other one is the "numerical gap" that consists in various steps of the retrieval process, such as image representation, distance measure, search strategy. To minimize this kind of gap, we have developed a compact and visually consistent image features, accurate and computationally inexpensive distance functions and efficient data structures for similarity search.

In [57], we propose an approach called HTM (Hierarchical Tree Matching) for the CBsIR problem. It consists of three main components:

1. a tree structure that models a hierarchical partition of images into tiles using color features;
2. an index sequence to represent the tree structure (allowing fast access during the search phase);
3. a search strategy based on the tree structures of both database images and the query image.

By using a tree to model the hierarchical decomposition of an image into tiles, our method is capable of handling virtually all parts of an image. Note that by using a

fixed decomposition we do not rely on image segmentation, typically not an accurate process. The number of partitioned tiles is fixed as long as the partition strategy is determined. The resulting tree is small for storage and speedy for searching. In addition, the parent-child relationship in the hierarchical tree structure implicitly facilitates the tile combination instead of using complex distance functions when matching images during the search phase. We store the image features associated with the nodes in the tree structure in the format of an index sequence, which allows fast access during the search phase. Also, we process the query sub-image by constructing a tree structure in the same way as the ones constructed for the database images, eliminating any size constraint on the query sub-image. The retrieval of relevant images is accomplished by efficiently comparing the query's tree structure with all the sub-trees of the database images. Then the distance between the tree structures can be effectively computed in order to rank the database images with respect to the query. Our experiments show that this strategy yields good results using different color features of the images, while consuming acceptable time and space. Compared to the related approach proposed in [39], our method is distinctly better based on the experimental results.

In the following, Section 3.1 presents the hierarchical partition of images and the tree structure to represent the decomposition. Section 3.2 and Section 3.3 provide an account of using different image features and their corresponding distance measures in our CBsIR system respectively. A brief review of the feature extraction method and its distance function in related work are presented in Section 3.4. Some practical considerations about efficiency and storage are listed in Section 3.5. Section 3.6 describes different strategies to search sub-images effectively. The experiments and results are discussed in Section 3.7. Finally, Section 3.8 concludes the chapter.

3.1 HTM's Hierarchical Partition and Tree Structure

To model an image, a grid is laid on it yielding a hierarchical partition and tiles. Although granularity could be arbitrary, we have obtained good results using a 4×4

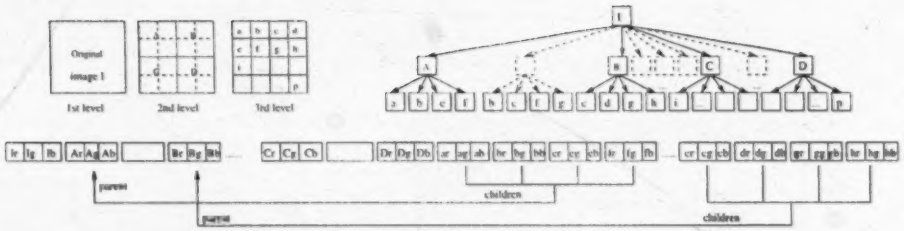


Figure 3.1: Hierarchical partition of an image with the resulting tree structure and one possible corresponding index sequence for storage.

grid resulting in a three-level multi-scale representation of the image (similarly to what was done in [28] and [39]). The hierarchical partition of an image with its resulting tree structure and one possible corresponding index sequence for storage (to be discussed in Section 3.5.2) are shown in Figure 3.1.

As illustrated in Figure 3.1, there are three levels in the hierarchical structure:

1. *The highest level*: the whole image itself.

2. *The second level*: the image is decomposed into 3×3 rectangles with each side having half the length of the whole image, yielding 9 overlapping tiles.

3. *The lowest level*: each tile of the second level is partitioned into 4 non-overlapping sub-tiles, resulting in 4×9=36 rectangles.

Note that to exclude redundancy in the CBsIR system, at the lowest level only the indices of the 4×4=16 unique tiles are stored with a small structure for relationship information. The features of the image tiles are associated to the nodes in the tree structures for images. Thus, every database image is represented as a series of tiles, each of which is mapped to a subtree of the tree structure modeling the image. Although similar, the tree model of the hierarchical partition is *not* the well-known Quadtree [5]. Our tree structure models the overlapping tiles at intermediate levels from the hierarchical partition, while the quadtree is used to describe a class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of non-overlapping spaces.

3.2 Feature Extraction using Average Color and Vectorial Distance

3.2.1 Average Color

As discussed in Section 2.1, color is one of the most prominent perceptual features to human and is commonly used in both academic and commercial image retrieval systems. To describe the color information of an image, the static and uniform quantization of a color space has well-known disadvantages (mentioned in Section 2.1) although it is the simplest scheme to reduce the number of colors present in an image.

An alternative to avoid this static quantization step is to reduce the color information by computing statistics about the color distribution. One of such statistics is the average color. Such methods have several advantages to be computationally simple, to result in very compact image feature descriptors, and to provide an efficient way for image comparison. On the other hand, of course, their effectiveness are sometimes compromised since images composed by completely different colors might yield identical statistics.

We use the average color of the image tiles in the RGB color space as one choice for image indexing in our CBsIR system. If the color components of a pixel P are P_R, P_G and P_B respectively, the average color for an image tile T is computed as:

$$\mathbf{U}_i(T) = \frac{1}{N} \sum_{P \in T} P_i \qquad i \in \{R, G, B\}$$

where N is the total number of pixels in the image tile T. Thus, a small three dimensional global color feature vector $V(U_R(T), U_G(T), U_B(T))$ is obtained per image tile.

3.2.2 Vectorial Distance Functions

Features alone cannot completely guarantee stability of the image retrieval system. Distance functions used to compare features also play an important role. An ideal distance function D and the feature $F(I)$ would satisfy the perceptual similarity:

$$D(F(I_1), F(I_2)) \text{ is small} \Leftrightarrow I_1 \text{ and } I_2 \text{ are perceptually similar.}$$

In most cases, visual features of an image are represented by high-dimensional vectors. These vectors can be treated as points in high-dimensional space (each vector element corresponds to a spatial coordinate). Therefore, it is natural to define distance functions in terms of Euclidean norms. The L_1 norm and L_2 norm as discussed in Section 2.1.3 are commonly used to compare two feature vectors. In practice, the L_1 distance function performs better than the L_2 distance function because the former is statistically more robust to outliers [3]. [10] suggests using a more complex quadratic form of distance measure which tries to capture the perceptual similarity between any two colors. That work uses low-dimensional color features as filters before using the quadratic form for the distance function, aiming to avoid intensive computation of quadratic functions. The advantages of modeling visual features in a vectorial space stand out. We can apply not only the computationally simple geometric distances to vector comparison, but also the spatial or metric access methods to speedup query processing if possible [53]. The use of access methods is important for large collections of images, because the query processing time should not increase in the same rate as the image collection increases.

Our distance measure for the statistics-relied feature extraction is based on the L_1 norm because it is simple and robust. After [39], the similarity of two feature vectors is determined by computing the weighted L_1-norm:

$$\| V(T_a) - V(T_b) \|_{db} = \sum_{i \in \{R,G,B\}} \frac{| V_i(T_a) - V_i(T_b) |}{\beta(V_i)}$$

where T_a and T_b are two different image tiles, V represents the statistics-relied three dimensional global color feature vector extracted from the image tiles, and $\beta(V_i)$ are the standard deviations of the respective features over the entire database.

Remember that we also need to pre-process the query sub-image in order to extract image features and build a data structure comparable with the tree structures of database images. Thus, the same hierarchical tree structure (Figure 3.1) is generated for the query sub-image as well as for all database images. Since we have the tree structures for both the database image and the query sub-image, we propose the following formula to compute the distance between the query sub-image Q and a certain tile I_S of a database image I (note that the full image is also considered a

tile of the image itself):

$$d(I_S, Q) = \frac{\sum_{i=1}^{m} \| V(I_{S_i}) - V(Q_i) \|_{db}}{m}$$

where m is the number of unique leaf nodes in the tree structure for a tile, and T_i and Q_i represent the corresponding leaf nodes in the tree structures for the tile and the query sub-image respectively. In effect, this is the average distance between the compared leaf nodes.

To measure the distance between a database image I and the query sub-image Q, we use a formula similar to the IHD method's [39] to obtain the minimum distance values among the comparisons of the query sub-image's tree structure with all the corresponding sub-tree structures of a database image. This image similarity measure is defined as:

$$DI(I, Q) = min_{i=0...NT_{db}} d(I_i, Q_j)$$

where $NT_{db} + 1$ represents the number of all sub-trees in the tree structure (tiles) of a database image that we should compare with the query's tree structure at different hierarchical levels, and j indicates the ordinal of the query's tree structure at a certain hierarchical level comparable to a sub-tree structure of the database image.

Figure 3.2 illustrates how we measure the similarity between the tree structures of the query sub-image and database images using the above idea. The root level of the tree structure represents the whole image. Using the previously discussed hierarchical partition (Section 3.1), the original image is progressively decomposed with a sliding window in order to capture overlapping image tiles. Figure 3.2 shows the full tree structure transformed from a 4×4 grid (There are 36 leaf nodes in the full tree structure.) Note that even though the query is a tile of the whole image, it has the same tree structure associated with it as the full image. Using the proposed distance measure, we can calculate the distance between the two tree as follows:

$$d(I_{full}, Q) = \frac{\sum_{i=1}^{16} \|V(I_i) - V(Q_i)\|_{db}}{16} = \frac{\sum_{i=1}^{16} \sum_{j \in \{R,G,B\}} \frac{|V_j(I_i) - V_j(Q_i)|}{\beta(V_j)}}{16}$$

Note that since all tiles of a database image is treated equally, no leaf node of the tree structures should be compared more than once. Hence only the 16 nodes corresponding to the 16 unique tiles are used in the comparison.

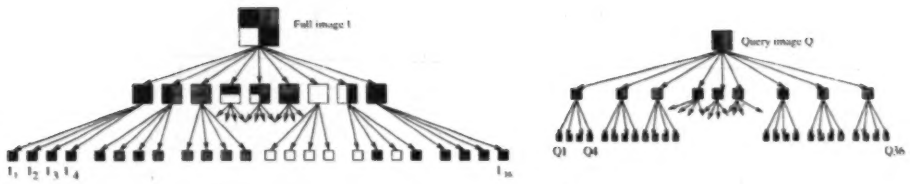


Figure 3.2: Simple example of tree structures for database image and query sub-image.

3.3 Feature Extraction and Distance Measure by Border/Interior Pixel Classification

Content-based image retrieval is performed based on abstract descriptions of the images extracted during the image analysis phase. Image analysis algorithms might depend on the properties of the images being analyzed, thus are usually distinct for different image domains and gradually change when the image domains expand. Unlike a narrow image domain which has a limited and predictable variability in all relevant aspects of its appearance, such as collections of fingerprints and X-rays of human skeleton, a broad image domain has an unlimited and unpredictable variability of the image's content. It is not possible to use semi-automatic techniques and domain-dependent knowledge during the analysis and comparison of images since the interpretation of the image's content is generally not unique and the image collections are very large as those formed by the huge amount of images available at the world wide web. In this scenario, low-level visual features of the images such as color and texture are especially useful to represent and compare images automatically.

In [54], a different alternative for CBIR in broad image domains is proposed. The authors propose the use of a simple yet powerful image analysis algorithm, whose result can be efficiently stored and compared without simplification avoiding the necessity of post-processing on the result of sophisticated image analysis algorithms used in region-based image retrieval approaches discussed before. This approach is called BIC (Border/Interior pixel Classification). The BIC method is made up of three main components: (1) a simple and powerful image analysis algorithm that classify image pixels as border or interior, (2) a new logarithmic distance

function to compare color histograms, (3) a compact representation for the visual features extracted from images. It is argued that the compactness, effectiveness and efficiency of BIC rely on its consistency among the analysis, representation and comparison of images.

The BIC approach has been shown to outperform several other CBIR approaches and, as such, we adopt it in our CBsIR system to extract the visual feature of each tile with the goal of improving the retrieval accuracy when compared with the simpler approach adopted in [57], where for each tile only the average color was recorded and used for image indexing. In this section, we focus on the discussion of image description within the BIC method proposed in [54]. The, we discuss BIC's distance function and compare it with other distance measures in Section 3.3.2. Section 3.5.1 will investigate the way how BIC achieves a compact representation of visual features extracted from images.

3.3.1 Image Description

The use of simple and robust ***image analysis*** algorithms, whose results can be preserved without approximation during the representation and comparison of the visual features, is the key to achieve efficient and effective CBIR systems in broad image domains. However, automatic segmentation algorithms have many drawbacks (as discussed in Section 2.2), which imply that they are very likely not the most adequate ones to deal with image retrieval tasks in broad image domains. The aforementioned reason spurs the proposal of a new image analysis algorithm in the BIC approach [54], trying to overcome these drawbacks from another point of view.

The BIC approach is based on a very simple (but powerful) image analysis algorithm that runs in time $O(n)$, where n is the size (pixels) of the image being analyzed. The image analysis algorithm in BIC uses the RGB color space uniformly quantized in $4\times4\times4$=64 colors. Any other color space and quantization scheme could be used as well, but this configuration is largely adopted in practice and seems to be a good uniform quantization scheme for the RGB color space [36]. The pixel count of each histogram bin is normalized between 0 and 255 for the sake of being able to represent a histogram bin using only one byte of memory. There is also no

clear advantage in using more than 256 distinct values per histogram bin as observed in practice. After the quantization step, image pixels are classified as either border or interior pixels. The classification criterion is: if a pixel is at the border of the image itself or if at least one of its 4-neighbors (top, bottom, left and right) has a different quantized color then it is classified as border pixel; if a pixel's 4-neighbors have the same quantized color then it is classified as interior pixel. Notice that this classification is mutually exclusive and it is based on a inherently binary visual property of the images.

The next step after pixel classification is to compute color histograms. Unlike the computation for global color histogram, here one color histogram is computed using only border pixels and another color histogram is computed using only interior pixels. In this way, each quantized color has the border/interior classification representation. In our CBsIR system, each image tile is thus described within BIC by means of two color histograms with 64 bins each (one for each quantized color). Assume an M-color model, a BIC histogram is an M-dimensional feature vector $(Bic_1^{class}, Bic_2^{class}, \cdots, Bic_M^{class}, class \in border, interior)$, in which each Bic_i^{class} represents the percentage of classified pixels in an image corresponding to each quantized color c_i. The BIC histogram Bic of an image tile T being of size $n_1 \times n_2$ is defined as:

$$Bic_{c_i}^{class}(T) = probability[p \in T_{c_i}^{class}] = \frac{\| T_{c_i}^{class} \|}{n_1 * n_2} \qquad class \in \{border, interior\}$$

where for any border or interior pixel p from image tile T, $Bic_{c_i}^{class}(T)$ gives the probability that the color of pixel p is c_i. $\| T_{c_i}^{class} \|$ is the number of pixels that are classified in either class with color c_i in image tile T.

The color coherent vector (CCV) approach of [12] also includes a binary classification of image pixels, which is nevertheless based on a non-binary visual property of the images - the size of the connected components. This requires the use of an empirical size threshold in order to have a binary classification in CCV. Most of the useful information about the size of the connected components are lost in this reduction and the approach may be very sensitive to the chosen threshold that varies according to the visual content of the images. Therefore, the CCV approach

is shown in [54] to be slightly more effective than a simple GCH.

BIC's classification of pixels in border/interior for each quantized color allows a more informed color distribution abstraction and is much more discriminative than a simple GCH or CCV according to [54]. This discriminative power can be analyzed for individual color in terms of texture, shape and connected components. For instance, the texture information of images in Figure 3.3 can be captured by the BIC classification in such a way that it yields distinct sets of color histograms for the two images, while the two GCHs are the same. The analysis of visual properties depends on the portion of the image covered by and also on the proportion between border/interior pixels for each quantized color. If the number of interior pixels for a given color is smaller than the number of border pixels for the same color, then at least one of the following visual properties could be a possibility: (1) the color is distributed in a relatively large areas with very irregular shape; (2) the color is distributed in small connected areas where the border of each area is larger than its interior; (3) the color is part of an image area that is rich in texture information. On the contrary, if the opposite situation is true, it can be concluded that (4) the color is distributed in relatively large and homogeneous areas with regular shape.

Figure 3.4[1] shows two examples of images analyzed by border and interior pixels. The original images are at the left column. The resulting binary images showing border pixels in black and interior pixels in white are at the middle column. The images showing border pixels in the corresponding original colors and interior pixel in white are at the right column.

3.3.2 dLog Distance Function

Apart from a simple and powerful image analysis algorithm, the BIC approach [54] also involves a new logarithmic distance (*dLog*) for comparing histograms. This *dLog* distance function has two main advantages over vectorial distances (e.g. L_1): (1) it is able to increase substantially the effectiveness of several histogram-based CBIR approaches, and at the same time, (2) it reduces by 50% the space requirement to represent a histogram. Now, we give a detailed study about the *dLog* distance

[1] From http://db.cs.ualberta.ca/mn/BIC/bic-sample.html

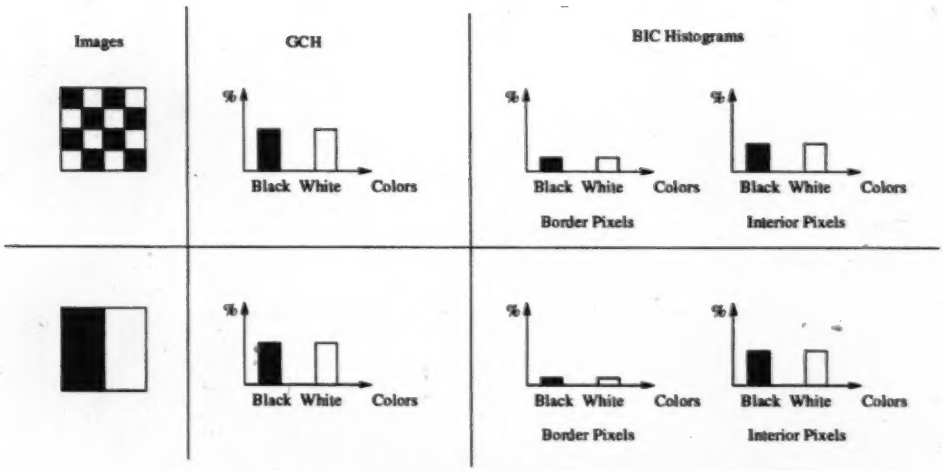


Figure 3.3: GCH vs. BIC Histograms for texture discrimination.

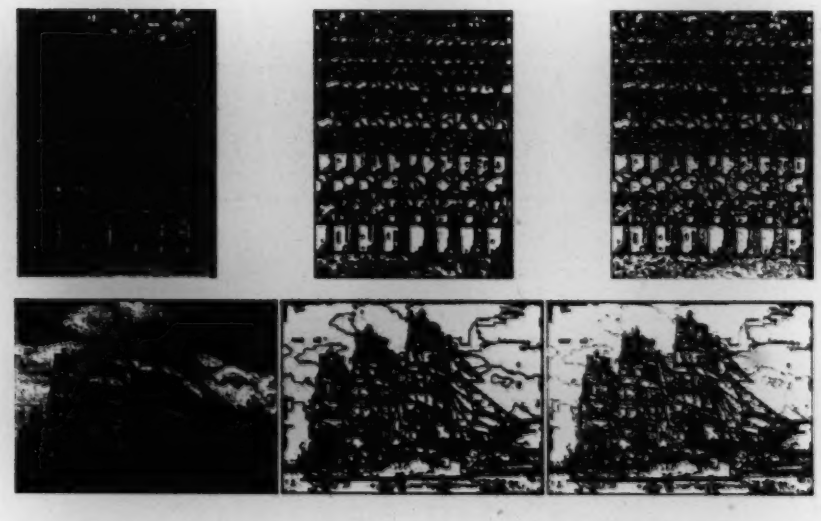


Figure 3.4: Two examples of the result by the BIC pixel classification.

function and how we accomodate it to our CBsIR system.

As discussed in Section 3.3.1, there are two color histograms with 64 bins each associated with each image tile. Actually, these two histograms can be stored and compared as a single histogram with 128 bins. Thus, any vectorial distance functions like L_1 or L_2 could be used to compare the BIC visual features. Although vectorial distances do have their advantages as mentioned in Section 3.2.2, they have also well-known limitations. One of such limitations is that a high value in a single histogram bin dominates the distance between two histograms, no matter the relative importance of this single value [36][50]. It is generally true that the foreground areas determines the semantic of the image and as such, it is more important to determine the similarity among images. On the other hand, it is equally true that, in general, the background covers the majority of the image area. Therefore, the tiles that compose the background are usually larger than the tiles that compose the foreground. For example, consider a set of images with a dominating and homogeneous background area of the image's content. Thus, this background can be represented in just one histogram bin. Now suppose we perform a similarity search using a query sub-image obtained from one of such images as example. When a vectorial distance is applied to compare these histograms, images having a background with the same color but a different foreground are retrieved ahead of any other image having the same foreground (a high degree of semantic similarity) but a background with a different color.

To deal with the above distortion based only on the information available within the histogram representation, the authors of [54] have proposed the *dLog* distance function which compares histograms in a logarithmic scale. The basic motivation behind this is based on the observation that classical techniques based on global color histograms treat all colors equally, despite of their relative concentration. However, the perception of stimulus, color in images in particular, is believed to follow a "sigmoidal" curve [50]. The relative increment in a stimulus is perceived more clearly when the intensity of the stimulus is smaller than when it is larger. For instance, a change from 10% to 20% of a color is perceived more clearly than an change from 85% to 95%. Indeed it has been a well observed phenomena regard-

ing many other phenomena involving how sensitive one is (including animals) to different stimuli [2]. Thus, the distance function is defined as follows:

$$dLog(a,b) = \sum_{i=0}^{i<M} |f(a[i]) - f(b[i])|$$

$$f(x) = \begin{cases} 0, & \text{if } x = 0 \\ 1, & \text{if } 0 < x \leq 1 \\ \lceil log_2 x \rceil + 1. & \text{otherwise} \end{cases}$$

where a and b are two histograms with M bins each. The values $a[i]$ and $b[i]$ represent the i^{th} bins of histograms a and b respectively. Here, M equals to 256 since the histogram bins are normalized between 0 and 255 as discussed in Section 3.3.1.

The *dLog* distance function does not solve the problem of comparing histograms when some bins are of very high values, but it diminishes this effects in most of the situations. In a log-scale, the range of distances between histogram bins becomes much smaller than in the original scale. For instance, the smallest distance between histogram bins in the original scale (being zero when both images have the same amount of a particular color) remains the same in log-scale. But the largest distance between histogram bins in the original scale (being 255 when the images have just one color and they are different) could be reduced to just 9 in log-scale, about 255/9=28 times smaller than in the original scale.

In [54]'s experiments, a study of substituting the *dLog* distance for L_1 in existing histogram-based approaches (e.g. GCH, CCV) shows that it clearly increase the effectiveness of all histogram-based approaches tested. The *dLog* distance function also plays an important role in the *BIC* histogram representation, which allows a substaintial reduction in storage. We will expand this issue in Section 3.5.1.

For the superiority of the *dLog* distance function, we accomodate it in our CBsIR system when using the BIC image feature description. Thus, the distance between two tiles T_a and T_b from images I_a and I_b respectively is defined as:

$$DT(T_a, T_b) = \frac{\sum_{i=1}^{m} dLog(H(T_{a_i}), H(T_{b_i}))}{m}$$

where T_{a_i} and T_{b_i} are sub-tiles of T_a and T_b respectively, represented as corresponding leaf nodes in the tree structures of the tiles, m is the number of unique leaf nodes in the tree structures at any hierarchical levels (if already at the leaf level, m=1),

and the H function computes the BIC histogram of each tile. The image similarity is measured similarly as in Section 3.2.2, which is based on the hierarchical tree matching scheme. The only difference is that here the BIC image description and the corresponding *dLog* distance function are used instead of the statistics-relied color features and the weighted L_1 distance function.

3.4 Feature Extraction and Vectorial Distances for Sebe et al's method (IHD)

3.4.1 Color Indexing

Related work [39] (called IHD method here) as discussed in Section 2.3 also uses color indexing for image feature extraction. Aiming to capture spatial relationships of color areas while also to preserve cheap memory cost and sufficient retrieval accuracy, the IHD method adopts the use of taking the covariance and the mean of the color distribution in a multidimensional color space [20] to index the image database. For the color features, the L*a*b* color space is chosen because it is perceptually uniform. The color features representing the color distribution include the average color $\mu = (\mu_L, \mu_a, \mu_b)$ and the covariance matrix $[\sigma_{ij}]$ $(i, j \in \{L, a, b\})$ of the color channels. If the color components of a pixel P are P_L, P_a and P_b respectively, then the index entries characterizing the color distribution of an image or an image patch A are:

$$\mu_i(A) = \frac{1}{N} \sum_{P \in A} P_i \qquad i, j \in \{L, a, b\}$$

$$\sigma_{ij}(A) = \frac{1}{N} \sum_{P \in A} (P_i - \mu_i(A))(P_j - \mu_j(A))$$

where N is the total number of pixels in the image or image patch A. Since the covariance matrix is symmetric, only 6 entries have to be stored. Hence, a nine dimensional global color feature $\nu_{color}(A)$ is obtained for the CBsIR system using the IHD method in [39].

3.4.2 Inter Hierarchical Distance (IHD)

For the IHD method [39], the full image I is also decomposed into a number of sub-patches by a similar partition strategy. While the partition-based approach used in [39] already introduces some spatial information, the space issue is explicitly considered leading to a new distance measure called *inter hierarchical distance* (IHD). The authors argue that the solution to extract the global features of image sub-patches $\nu_{color}(A)$ in order to represent the spatial information would increase the index size dramatically. However, if only the differences of the global features of the image and its sub-patches are stored, then the spatial encoding is guaranteed without a major increase of the index size. Thus, a measure of the distance between the global features of the image and the features of its sub-patches is proposed. This distance is called *inter hierarchical distance* (IHD) since it is taken between feature vectors of different hierarchical levels of the image partition.

In case of color features discussed in Section 3.4.1, a two dimensional feature vector is used. The vector components are the L_1-norm of the differences of the mean and covariance elements respectively [39]:

$$V^l_{IHD,1}(A) = \sum_{i=L,a,b} |\mu_i(A) - \mu_i(A_l)|$$

$$V^l_{IHD,2}(A) = \sum_{i,j=L,a,b} |\sigma_{ij}(A) - \sigma_{ij}(A_l)|$$

where A is the full image and A_l is a certain sub-patch, μ is the mean element and σ is the covariance element from the $L*a*b*$ color space. However, there is no such tree-like data structure for the IHD method. So for the example in Figure 3.2, the distances between the query and the full database image as well as its sub-patches S are computed as follows:

$$d(I_S, Q) = \|V^S_{IHD}(I) - V^Q_{IHD}(I)\|_{db}$$

Note that when $S = 0$, I_S represents the full image, and thus $V^0_{IHD}(I) = 0$.

An importance difference between IHD and HTM is that instead of considering only the global feature information represented by IHD vectors of the query sub-image and a certain sub-patch in a database image, our HTM method also uses the

local information of a tile represented by the leaf nodes in its tree structure. The average of distance values among the corresponding leaf nodes is regarded as the distance between the tree structures of query sub-image and a certain tile of the database image at any hierarchical level. We will demonstrate in our performance study that the HTM scheme using statistics-relied feature extraction results in much more retrieval accuracy with small extra query processing time cost when compared with the IHD method, although much more information of the hierarchical structure needs to be stored and compared.

Since the tree structure for the query sub-image could also be a sub-tree of the tree structure for the database images, we float the tree structure of the query sub-image within the whole tree structure of database images using the search strategy presented in Section 3.6.1. When comparing the tree structures of query sub-image and the tiles of database images with statistics-relied color features, we apply the distance measures presented in the previous sections to compute the distances between them.

3.5 Efficiency and Storage Consideration

As image databases grow larger, image retrieval systems need to address efficiency issues in addition to the issue of retrieval effectiveness. Efficiency concerns lie in every phase of the retrieval process. In this section, we focus on investigating the methods that improve the efficiency and the compactness of image indexing, without compromising effectiveness. Section 3.6 will discuss efficiency concerns in the search phase afterwards.

3.5.1 Feature Representation

As discussed above, our CBsIR system extracts features of the image database in two distinct ways. For one statistics-relied feature extraction, the color feature representing the average color of each tile of the database image is just a small three dimensional feature vector. This feature extraction method has the inherent advantage to result in very compact descriptors that are easy to compute and efficient for

searching. Although the effectiveness of this method might be affected sometimes (e.g., as mentioned in Section 3.2.1), the search strategy based on the hierarchical tree matching scheme (discussed in Section 3.6.1) would remedy this in most cases.

Also statistics-based, the BIC method of [54] is adopted in our CBsIR system for image feature extraction. We have already discussed two of its components - a simple yet powerful image analysis algorithm and a new logarithmic distance for histogram comparison in Section 3.3.1 and Section 3.3.2 respectively. Now we study in detail the third component of BIC which yields a compact representation of the image visual features allowing efficient image comparison.

When the $dLog$ distance function is used to compare histograms, it is possible to store the result of the $f(x)$ function instead of the normalized pixel count. The comparison of the histograms according to the $dLog$ distance thus becomes computationally simpler. A more careful look at the definition of the $dLog$ function reveals that it is in fact an L_1 distance of the log of the pixel count - $f(x)$. Therefore, all we have to do is just compare the log-based represented histograms using the L_1 vectorial distance.

Besides, remember that in $f(x)$ function x stands for each bin of a histogram whose value range is between 0 and 255. Therefore, the $f(x)$ can be perceived as an integer between 0 and 9. It can assume only 10 distinct values and these values can be stored in just 4 bits ($10 < 2^4$). This means that the log-based representation of histograms requires only half of the space necessary to store the normalized pixel count which is the original representation.

The log-based representation allows a reduction of 50% in the required storage space for any histogram-based CBIR approach [54]. For the particular case of the BIC approach, it is possible to store a BIC histogram being of 128 bins (64 for border pixels and 64 for interior pixels) in just 64 bytes of memory. This is a very compact representation of image visual features. Thus, high-end workstations can maintain fairly large collections of images in memory, completely avoiding the need of disk-based access methods to speedup query processing.

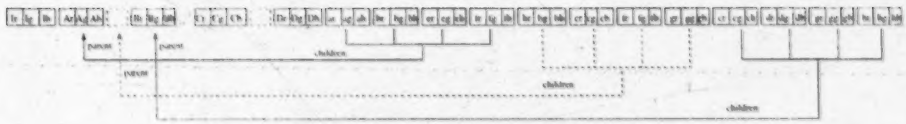


Figure 3.5: Index sequence.

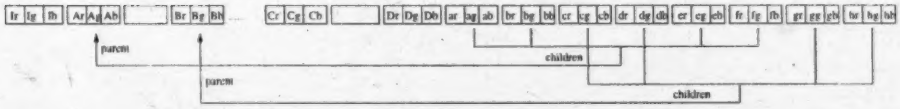


Figure 3.6: Index sequence without redundant tile at the lowest level of the hierarchical structure.

3.5.2 Index Sequence

Apart from compact image representations in our CBsIR system, we also consider a compact storage format for the visual features that allows fast access during the search phase.

From the progressive decomposition strategy (illustrated by the hierarchical structure in Figure 3.1), a predefined parent-child relationship for the tree structure can be easily extracted. Using the tree structure introduced in Figure 3.1, Figure 3.5 gives an illustration of the *index sequence* representing such relationships.

In Figure 3.5, each node in the tree structure is represented by the sequence for the elements of the three dimensional feature vector based on statistics-relied image indexing or by that for the bins of the BIC histograms. The relationship of parent node and child nodes in the tree structure is maintained by a predefined order of sequences in the index.

Note that Figure 3.5 represents the storage of redundant tiles at the lowest level of the hierarchical structure. Since this index sequence is stored on secondary storage and aimed for fast retrieval, we apply an immediate improvement (shown in Figure 3.6) by storing only the indices of the 16 unique tiles at the lowest level (excluding the redundancy) with a small structure of the relationship information as an extra overhead. This cost is much less compared with that of storing and fetching the information about those redundant tiles, which would thus further speedup the image comparison in search phase.

3.6 Search Strategy

Searching is a fundamental problem in computer science [53]. However, similarity search in digital images that are close or similar to a given visual query is inherently different from the exact-match search in traditional database systems. Apart from introducing the use of filtering techniques and access methods to reduce the complexity of the searching process, some approximate methods relying heavily on clustering techniques to classify similar objects together are also applied for the indexing of non-metric spaces in such case that the precision of a query can be relaxed to reduce the query processing time. In the following, we discuss in detail our search strategy for sub-image search.

3.6.1 Search by Hierarchical Tree Matching

The search algorithm in [51] uses an expensive branch-and-bound procedure to retrieve the best match, preserving the query's scale. The IHD approach in [39] simply follows a linear scan (as "sequential scanning" in Section 2.1.4) to compare the IHD vectors of the query sub-image and database image patches, which achieves fast speed (because of the compact feature representation being just a two dimensional vector) but compromises accuracy as we will discuss in our performance study. Here, we combine the above approaches and come up with a hybrid of both so as to expediate the matching process without compromising effectiveness. Besides, this search strategy is based on a distinct query representation: unlike the scale-preserving sub-image retrieval in [51], we generalize the image comparison by constructing the same tree structure to represent the hierarchical partition for both the database images and query sub-image, so that images can be retrieved independent of the size of the sub-image (i.e., scale independency). A formal algorithm written in pseudo code (Figure 3.7) summarizes the search strategy of the HTM method for finding the most similar images to the sub-image query within the whole database.

To further illustrate the search strategy, Figure 3.8 shows the process of finding the best matching image and the updating of the tree structure of the query sub-

```
procedure  SearchHTM(databaseMetafile, queryTree)
begin
1   for each  image I in the database  do {
2.    dbImageTree := fetchMeta(databaseMetafile, I);
3.    entry := 0;
4.    tempDist[entry++] := fullTreeCompare(dbImageTree, queryTree);
5.    for each  subsequent level L of the hierarchical structure  do {
6.      querySubtree := updateQuery(queryTree, L);
7.      for each  subtree I_sub at level L of the hierarhical structure for Image I  do {
8.        tempDist[entry++] := subTreeCompare(dbImageTree, I_sub, querySubtree);
9.      }
10.   }
11.   distArray[I] := findMinimum( tempDist);
12. }
13. rankList := sortForMinimum(distArray);
end
```

Figure 3.7: Search algorithm of the HTM method.

image in the search phase using the example given in Figure 3.2.

First, in step (a), we compare the full tree structures of the database image I and the query sub-image Q. Note that, at this point, the query sub-image has the same tree structure (a three-level hierarchical structure) as the database image. As discussed in Section 3.5.2, the tree structures are mapped into index sequences, which maintain the relationship of nodes inside the tree structures. Each piece in the index sequence stores the information about the image feature associated with a certain node in the tree. Using the distance measures discussed for different methods and feature representations respectively, we can obtain the distance between the full trees.

In order to compare the query with the sub-tiles of the database image, we float the query sub-image's tree structure within that of the database image. Before doing

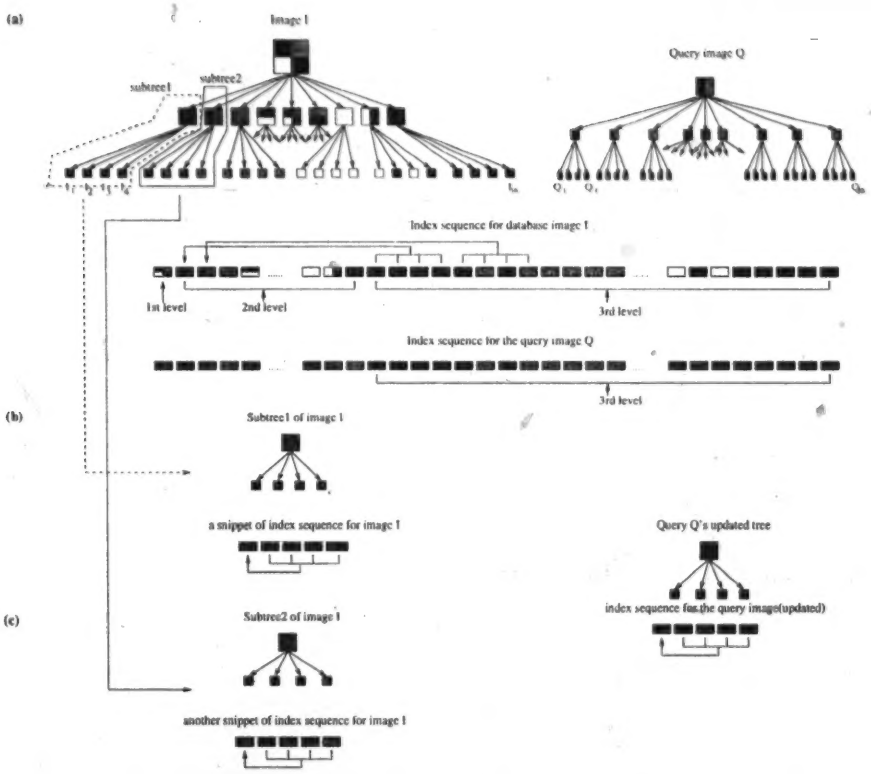


Figure 3.8: Determining the best matching image (sub)tree.

this, we have to update the tree structure of the query image. Since we want to make the query's tree and the database image's sub-trees (e.g., subtree1 of image I in Figure 3.8) comparable, we have to reprocess the query sub-image as to obtain a tree structure similar to a certain database image's sub-tree at a particular hierarchical level. In Figure 3.8, the new query tree is shown to the right of steps (b) and (c). The indices of the leaf nodes in the query's updated tree structure is used to continue comparing with the sub-trees of the database image. For this example, step (b) shows the comparison of the query sub-image with the first sub-tree of the database image I. In step (c), the second sub-tree of the database image (subtree2 of image I in Figure 3.8) is compared with the query's new tree. (According to our distance measure, subtree2 is determined as a perfect match for the query.) The comparison is similar for the remaining sub-trees and those at the lower levels of the database image's tree structure. Finally, the minimum distance representing the best

matching sub-tile is used as the distance for the database image between the query. In short, the whole searching process is done by updating the query's tree structure and floating it around inside the full tree structure of the candidate database image for tree comparison at different hierarchical levels, until there are no more sub-trees left.

Experiments detailed in Section 3.7 show that this search strategy by the hierarchical tree matching scheme yields better retrieval accuracy compared to related work (e.g., [39]) at the cost of small storage overhead. The query processing time cost is also very acceptable compared to [51]. Moreover, it should be emphasized that this search strategy has implicitly solved the *object localization* problem (an adjunct issue to the sub-image retrieval problem) along with the search task. By using the hierarchical partition strategy, the spatial information is actually kept in the relationship of tiles represented in the hierarchical tree structure. During the process of determining the distance between the query sub-image and the candidate database image, the best matching tile that yields that minimum distance from the query among all sub-tiles and the full tile of the database image is filtered out, which also implies the location information of this tile as to where it is located inside the full image because of the hierarchical decomposition. Note that, in this way, we do not need any additional process to deal with the object localization problem, which suggests the superiority of our search strategy by the hierarchical tree matching scheme.

3.7 Performance Study

3.7.1 Performance Measures

Our sub-image retrieval task is to retrieve, as highly ranked as possible, the image from which a given sub-image was extracted. Hence, for each query sub-image there is only one relevant answer, namely the original image. We apply the following two measures (also used in [25][39], etc.) to evaluate the effectiveness of various competing sub-image retrieval approaches. If $Q_1, \cdots, Q_n$ are query sub-images, and for the i^{th} query Q_i, I_i is the only image that "contains" Q_i, (i.e., Q_i

"appears" in I_i). A method is said to be better if it has lower *average r-measure* and a higher *average precision*. Since there is only one relevant image per query, traditional Precision × Recall graphs are not meaningful here.

1. *Average r-measure* gives the mean rank of the correct answer averaged over all queries: $\frac{1}{n}\sum_{i=1}^{n} rank(I_i)$.

2. *Average precision of a method* gives the average of the precision values over all queries' recall points (with 100% being perfect performance), i.e., at the correct answer: $\frac{1}{n}\sum_{i=1}^{n} \frac{1}{rank(I_i)}$.

Now for the performance study, we use two kinds of criterion. One is the effectiveness, which is measured via *average r-measure* and *average precision*. The other criterion is efficiency, which we evaluate by the space and time requirements of the compared approaches.

Initially, we measure the distance between the database images and the query sub-image using a default precision of 6 decimal digits. But it becomes clear that it is not appropriate. The distance between many images would differ only in the 5^{th} or even 6^{th} decimal digit. Since there are so many approximations in the retrieval process, e.g., the image partitioning, the use of statistics-relied feature vector per partition, etc., it does not seem to make sense to use such a fine granularity for the distance calculation. In addition, humans do not have such a fine distinction between perceived similarities. Thus, we decide to use only two decimal digits precision. An immediate consequence of this lower distance granularity is the increase in the number of tied distances. Let us call the set of images with the same distance a *group*. As we shall see shortly this can have a large impact on the results depending on how one defines and measures the rank of a relevant image.

We adopted two kinds of measurement to rank retrieved images [57]. One is the *average actual rank*, which is the average between the minimum and maximum ranks for the images inside the same group as where the relevant image is. Assume $rank(I_j)$ is the absolute rank of image I_j after ordering all images by their distance to the query image (with ties broken arbitrarily). Then if a relevant image I_j has the same distance as images $I_i, I_{i+1}, \cdots, I_k (i \leq j \leq k)$, the average actual rank of

I_j is defined as $(min_p(rank(I_p)) + max_p(rank(I_p)))/2, p \in \{i, \cdots, k\}$. The other measure is the *group rank*, where all images inside the same group have the same rank which is the rank obtained as if the whole group was a single "object". In this case an image's rank does not depend on its group size, but rather on how many groups of images rank before the image's own group. This reflects the fact that if two images have the same distance, they should also have the same ranking, as any difference in ordering is only "accidental". Table 3.1 exemplifies the measures above. Note that while the group rank is quite optimistic, the average actual rank is probably more realistic.

Table 3.1: Average actual rank and group rank.

Image	Distance	Original Rank	Average Actual Rank	Group Rank
I_A	0.05	1	1.5	1
I_D	0.05	2	1.5	1
I_F	0.43	3	4	2
I_T	0.43	4	4	2
I_M	0.43	5	4	2
I_B	0.67	6	6	3

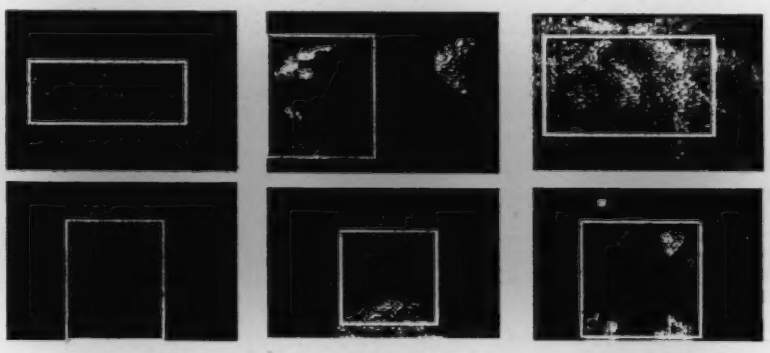
3.7.2 Experimental Setup

It is important to evaluate performance scientifically so as to ensure the validity of the results. In order to test the robustness of different feature extraction methods and the learning aptitude of the further improved CBsIR system with the relevance feedback technique discussed in Chapter 4, we use an image database with 10,150 images: a mixture of the public Stanford10k[2] image dataset and some images from one of COREL's CD-ROMs. The image database Stanford10k contains color JPEG images of size 128×85, 85×128, 128×96, or 96×128, etc. The database images have the same dimensions, but not necessarily the same orientation. Our well-balanced large-scale testbed is very realistic and helps us reach a fair evaluation of different methods. The content of the database images ranges from animals, people, scenery, and architecture, etc.

[2]http://www-db.stanford.edu/~wangz/image.vary.jpg.tar.

For the query sub-image datasets, we have constructed two different query sets to evaluate different aspects of the system. Both of them are obtained by manually cropping part of the original images. These original images are considered as the *unique correct* answer for the respective queries. The first query set consists of 20 query images ranging over different themes. A sample of the query sub-images, along with those from which they are extracted are displayed in Table 3.2. The size of the query sub-images varies, being on average 33% the size of the database images. The performance results are collected on a computer running Linux 2.4.18 with two AMD Athlon MP 2400+ CPUs and 2GB of main memory. As for the second query set, we obtain 21 query sub-images which are distinct from those in the first query set. This is because we want to test how accurate the combinations of the HTM scheme with different images features, such as average color and the BIC histograms, could hit the original image by distinguishing it from the similar images that belong to the same category as that of the answer image (original image). The size of the query sub-image also varies, being on average of 18% the size of the database images. Experimental results on this database are collected from the online demo[3] on a computer running Linux 2.4.17 with two Pentium III CPUs and 256MB of main memory. The above data serve as the ground truth to test different image features and retrieval methods.

Table 3.2: Six sample images with query sub-images indicated by the white frames.



We apply three different image features: the statistics-relied feature (average

[3]http://db.cs.ualberta.ca/mn/CBsIR.html

color), the BIC histogram, and the IHD feature vector on the above image database. And we compare the performance of the combinations of the HTM scheme with different image features, as well as the HTM-based retrieval method versus other related approaches. Besides, the average size of query sub-images is another tuning factor in the experiments.

Specifically, we use the first query set to obtain a preliminary comparison of the HTM scheme using average color feature of image tiles and the corresponding distance measures with the IHD method. For our HTM method, the RGB color space is used while for the IHD method, the L*a*b* color space is applied instead. As for the second query set, we apply the average color feature vectors and the BIC histograms on the HTM scheme and compare these combinations with the IHD method. To compute the BIC histograms, we consider the RGB color space with quantization into C colors. The tradeoff between retrieval accuracy and efficiency using the BIC methods is studied by changing this parameter.

3.7.3 Results Analysis

First, we compare our HTM method with related work using the first query set. Table 3.3 shows the effectiveness obtained using the *average actual rank* (defined in Section 3.7.1). Here, the statistics-relied image feature representation (*average color*) is chosen for the reason that the compared IHD method also uses the average color but with an additional covariance matrix to represent the color distribution. Apparently, the color feature representation of the IHD method is more complex and more informative than HTM's statistics-relied feature representation. However, Table 3.3 clearly demonstrates the superiority of HTM's retrieval scheme. When looking at the *group ranks* (defined in Section 3.7.1) in Table 3.4, even though HTM is still superior, a relatively much better result is obtained for the IHD method. The average r-measure of 1030, obtained when using the average actual rank, dropped drastically to 26 using the group rank. This *misleading* result is due to the large number of ties inside the groups obtained for the IHD method. Using the IHD method, we obtain groups as large as 351 images, each group having 83 images on average. On the other hand, the HTM method is able to discriminate better. No

Table 3.3: Performance of different methods using average actual rank.

Methods	HTM/Avg	IHD/AvgCov
Avg r-measure	31.2	1030
Avg precision	0.34	0.01

Table 3.4: Performance of different methods using group rank.

Methods	HTM/Avg	IHD/AvgCov
Avg r-measure	10.45	26
Avg precision	0.36	0.2

group is larger than 31, and on average each group contains about 4 images. That is, the less discriminating an approach is, the more ties it will yield. And the more ties there are, the less groups exist. Hence, the lower the group ranks. Nevertheless, the more precise and discriminative an approach, the closer the two rank measures.

A more interesting conclusion can be drawn if one uses only the queries that yields the 10 best results for each method. Using the average actual rank, our HTM method yields 2.5 for average r-measure and 0.65 for average precision while the IHD method obtains 191.7 for average r-measure (a very large improvement when compared to the 1030 obtained for all 20 queries) and 0.02 for average precision. Using group rank, our HTM method produces 2.4 for average r-measure and 0.65 for average precision, while the IHD returns 5.7 for average r-measure and 0.37 for average precision, another large improvement. One should note that these figures for the IHD method are quite close to the ones reported in [39] - even though we use a different dataset and different queries - which suggests to us that that paper may have used the group rank as a measure of retrieval effectiveness.

Because of the distinct difference between the two methods for the aspect of effectiveness, it is worthwhile to look at the distributions of the ranks of the relevant (original) images. As can be seen in Table 3.5, for our HTM method 80% of the relevant images are ranked among the top 50 retrieved images, while for the IHD method only 10% of the original images are ranked among the same top 50. Examples of sample queries and answers by different methods are shown in Figure 3.9. Furthermore, Table 3.6 gives the top 3 retrieval results for six sample query sub-

Table 3.5: Comparison of the distributions of relevant images using average actual rank.

	Number of Queries(%)	Number of Queries(%)
Rank Range for Original Images	HTM/Avg	IHD/AvgCov
[1,10]	12 (60%)	1 (5%)
[11,20]	0 (0%)	1(5%)
[21,50]	4 (20%)	0 (0%)
[51, 100]	3 (15%)	2 (10%)
[101,500]	1 (5%)	5 (25%)
[501, 5000]	0 (0%)	11 (55%)

images and their corresponding original images' ranks using average actual rank by the HTM method.

Table 3.7 shows the average cost, measured in seconds, to process a query, i.e., to access all metadata, obtain the distance between the sub-image query and the database images and sort the resulting file. There is a slight difference between the two methods for the time cost of search phase. The IHD method is faster than our HTM approach since it only stores a two dimensional feature vector per each tile of the database images and applies a simple linear search on them. Even though the IHD method is 33% faster than HTM, it is important to note that HTM processes a query using much more hierarchical structure information in the tree matching phases and deals with a database of 10,150 images, where all the metadata is stored on disk, *not* in main memory, still very fast, namely in 0.15 seconds on average. As we have pointed out before in Section 3.4 that the improved effectiveness by the HTM method clearly dominates the small extra time cost comparing with the IHD approach, which is very acceptable.

In order to extract image features from the image database and generate the metadata file, our HTM method use 3.35 hours while the IHD method use 4.31 hours using the machine mentioned in Section 3.7.2. (Note that this procedure can be done off-line). When looking at the space cost for those disk-resident index files, the HTM method would require 4.38 MB while the IHD method would need less storage, namely 3.25 MB. Here, our implementation stores the indices of all the

Query sub–image

HTM: 1; IHD: 14

Query sub–image

HTM: 3; IHD: 53

Query sub–image

HTM: 5; IHD: 317

Figure 3.9: Sample query sub-images and their original images (answers) with average actual ranks using different methods. (Lower ranks are better.)

Table 3.6: Search results for six sample query sub-images by HTM. (Note that the size of query sub-images and retrieved images is changed for viewing purpose.)

Query Sub-Image/ Rank of Relevant Image	Top 3 Retrieved Images
/1	
/1	
/2	
/3.5	
/40.5	
/51	

Table 3.7: Comparing query processing efficiency for HTM and IHD.

Methods	Search Phase (sec.)
HTM/Avg	0.15
IHD/AvgCov	0.10

leaf nodes (ignoring the intermediate nodes) for the partition strategy, which means there is still much duplicated information because of the overlapping at the second level in the hierarchical structure. That is why the index file takes more space than the IHD index file. However, considering the much better effectiveness of our HTM method compared to the IHD method, this extra cost seems worthwhile. In addition, as discussed earlier, this is an issue which can be improved using a more storage-conscious implementation.

In a word, based on our experimental results, our proposed HTM method is very effective compared to the IHD method of [39] with very acceptable retrieval time and space cost. Although it is not completely clear why there are a few outliers in the results, we believe the low contrast between the tile of interest and the image's background is to be blamed. Another possible source of problem seems to be images with an unusual large number of colors, detracting the discriminative power from the average color feature.

One venue to further explore the HTM scheme is to try using more powerful yet compact representation for the tile features. Now, we apply the BIC approach discussed before as well as the statistics-relied approach for image feature extraction on the database. Here, the second query set is used to test the robustness about the HTM scheme when there are several images that are similar to the correct answer of the query sub-image. Also, the related IHD method is applied on the second query set, serving as a comparison. For experiments on this query set, we use the average actual rank measure since it seems more realistic.

Table 3.8 gives the retrieval accuracy comparison using different image feature representations with the HTM scheme and the IHD approach. For the BIC method, the BIC parameter (SIZE) refers to the number of uniformly quantized colors on the RGB color space. The average color is also extracted from the RGB color

Table 3.8: Retrieving original images using IHD and different feature representations with HTM.

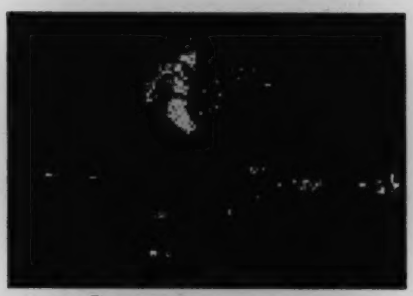
Methods	HTM/BIC	HTM/BIC	HTM/Avg	IHD/AvgCov
BIC parameter(SIZE)	64 colors	16 colors	-	-
Avg r-measure	2.24	22.81	573.29	1360.90
Avg precision	0.91	0.43	0.15	0.002

space, which is the original approach used in [57]. From Table 3.8 we can see that the IHD approach produces the worst results, which is consistent with what we have obtained from the experiments using different query set. In addition, even using just 16 quantized colors for image feature representation by BIC, the system achieves better retrieval accuracy than using the average color for image feature representation. And the use of 64 quantized colors yields very good results and is the best among all. For a more detailed comparison, Table 3.9 also shows the distributions of the ranks of the original images using different image feature representations with the HTM scheme and the IHD approach.

Besides, it should be noticed that the average size of the query has some effect on the final retrieval accuracy. When the average size of the query decreases, using simply the average color of each tile could not achieve good performance since it is far from discriminative than using BIC. The values for the HTM/AvgColor combination are better in [57] because the average query size is then larger than the one (being 18%) used in the experiments reported here. Because query sub-images and the database images have similar resolution (result of the cropping operation), larger query has more details about the object of interest it embodies, thus can be more discriminative. If we were to use larger query sizes for both image feature representations with the HTM scheme, the relative advantage of HTM/BIC would be even larger. Table 3.10 compares the average cost of query processing time for the use of different BIC histograms for feature extraction. Some sample queries and their answer images' ranks retrieved from the above database using the IHD approach and different image indexing methods with the HTM scheme are displayed in Figure 3.10 as well.

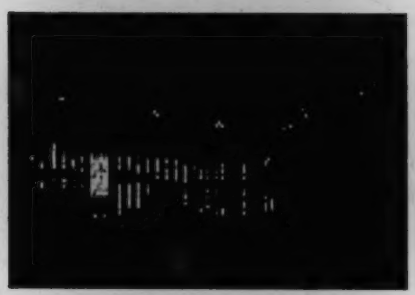
BIC/64colors: 1
HTM/Avg: 33

BIC/16colors: 1
IHD/Avg+Covariance: 2396



BIC/64colors: 1
HTM/Avg: 132

BIC/16colors: 8
IHD/Avg+Covariance: 322



BIC/64colors: 1
HTM/Avg: 314

BIC/16colors: 18
IHD/Avg+Covariance: 410

Figure 3.10: Sample query sub-images and answers with ranks using the IHD method and different image indexing methods with HTM. (Lower ranks are better.)

Table 3.9: Comparison of the distributions of original images using different feature representations.

Methods	HTM/BIC	HTM/BIC	HTM/Avg	IHD/AvgCov
BIC parameter(SIZE)	64 colors	16 colors	-	-
Rank Range	No. of Qs	No. of Qs	No. of Qs	No. of Qs
[1,10]	20 (95.2%)	14 (66.7%)	5 (23.8%)	0 (0%)
[11,20]	0 (0%)	3 (14.3%)	5 (23.8%)	0 (0%)
[21,50]	1 (4.8%)	0 (0%)	2 (9.5%)	0 (0%)
[51, 100]	0 (0%)	2 (9.5%)	0 (0%)	0 (0%)
[101,500]	0 (0%)	2 (9.5%)	3 (14.3%)	6 (28.6%)
[501, 5000]	0 (0%)	0 (0%)	6 (28.6%)	15 (71.4%)

Table 3.10: Comparing query processing efficiency using different BIC histograms with the HTM scheme.

Methods	Search Phase (sec.)
HTM/BIC/64 quantized colors	1.89
HTM/BIC/16 quantized colors	0.56

3.8 Summary

In summary, we have proposed a new method called *Hierarchical Tree Matching* (HTM) for the problem of content-based sub-image retrieval (CBsIR). The highlights of the HTM method are:

1. it adopts a multi-scale hierarchical partition to model both database images and the query sub-images in trees, eliminating any reliance on the typically complex and inaccurate image segmentation as well as any size constraint on the query sub-image;

2. it stores the image visual features associated to the tree structure in the format of an index sequence, allowing fast access during the search phase;

3. it uses a distance measure that considers not only the global feature information of a sub-tree structure but also the local spatial distribution offered by the child nodes in the sub-tree structure so as to capture more detailed distinction;

4. it applies a search strategy that compares the tree structures of both database

images and query sub-image at any hierarchical levels for the best matched tile and locates the object of interest at the same time.

Our experimental evidence shows that this method outperforms the recently proposed partition-based CBsIR method [39] by achieving a good balance of retrieval accuracy and efficiency. As a side contribution, we have shown that one can obtain very different rank measures depending on the distance granularity and ranking criteria in the presence of ties.

We have also studied different methods for image indexing with the HTM scheme, such as the statistics-relied average color feature vector approach and the BIC method [54]. The extraction of average color feature from each tile as the statistics-relied approach is very easy to compute and yields very compact descriptors for efficient image comparison, but has limited discriminative power in similarity matching. Whereas, based on a simple yet powerful image analysis algorithm and a new logarithmic distance function, the BIC method achieves not only compact representation for the visual features extracted from image tiles but also outstanding accuracy in retrieving images of broad image domains. It is clear that the combination of the proposed HTM scheme with image indexing by the BIC method has notable superiority for CBsIR.

Chapter 4

Supervised Learning in Content-Based Sub-Image Retrieval

We have already shown that the HTM method is a stable, effective and efficient approach for content-based sub-image retrieval. It is, however, impossible for any image retrieval method to be entirely "foolproof". The main reason is the gap between low-level image features and semantic contents of images. This problem arises because visual similarity measures, such as color histograms, do not necessarily match the *semantics* of images and human *subjectivity*. Human perception of image similarity is subjective and task-dependent, that is, people often have different semantic interpretations of the same image. Even the same person may perceive the same image differently at different times. In addition, each type of visual feature tends to capture only one aspect of the image property and it is usually hard for a user to specify clearly how different aspects are combined to form an optimal query. Therefore, for any query sub-image, not all top ranked images retrieved by a retrieval method are actually relevant according to the user's perception. To address this problem, interactive relevance feedback techniques have been proposed to incorporate human perception subjectivity into the retrieval process. Users can thus be prompted to evaluate the results by marking each retrieved image as "relevant"or "irrelevant". Queries or similarity measures are automatically refined on the basis of these evaluations, which potentially improves the quality of retrieval.

In this chapter, we investigate the use of information, provided interactively by a user, to improve the performance of the HTM-based approaches for CBsIR.

Inspired from techniques in RBIR [46][56], we outline a tile re-weighting scheme that uses feedback information (in the form of labeled examples). Learning is thus effected by interpolating the query vector with feature vectors of positive examples. Furthermore, the feedback information is also incorporated into the image similarity measure based on the *tile re-weighting* scheme, implicitly refining the final ranking of retrieved images.

For organization, Section 4.1 discusses some research work in other related image retrieval situations. Then the learning method for CBsIR are introduced in Section 4.2. The performance measures (Section 4.3), the experimental framework and results (Section 4.4) are presented next. Section 4.5 closes the chapter with a brief summary.

4.1 Relevance Feedback in Other Image Retrieval Scenarios

4.1.1 Learning in traditional CBIR

The key issue in relevance feedback is how to use positive and negative examples to refine the query and/or to adjust the similarity measure. Early relevant feedback schemes for CBIR are adopted from feedback schemes developed for classical textual document retrieval. These schemes fall into two categories: query point movement (query refinement) and re-weighting (similarity measure refinement), both based on the well-known vector model.

The query point movement methods aim to improve the estimate of the "ideal query point" by moving it towards positive example points and away from the negative example points in the query space. One frequently used technique to iteratively update the query is the Rocchio's formula [1]. It is used in the MARS system [18], replacing the document vector by visual feature vectors. Another approach is to update query space by selecting feature models. The best way for effective retrieval is argued to be using a "society" of feature models determined by a learning scheme since each feature model is supposed to represent one aspect of the image content more accurately than others.

Re-weighting methods enhance the importance of a feature's dimensions that help retrieve relevant images while also reduce the importance of the dimensions that hinder the process. This is achieved by updating the weights of the feature vector in the distance metric. The refinement of the re-weighting method in the MARS system is called the standard deviation method [18]. Another alternative for learning the distance metric is to automatically select the best one from a set of predefined distance metrics for the retrieval process based on the relevance feedback, e.g., [21].

Recent work has proposed more computationally robust methods that perform global feature optimization. The MindReader retrieval system [26] formulates a minimization problem on the parameter estimating process. Using a distance function that is not necessarily aligned with the coordinate axis, the MindReader system allows correlations between attributes in addition for different weights on each component. A further improvement over the MindReader approach [37] uses a unified framework to achieve the optimal query estimation and weighting functions. By minimizing the total distances of the positive examples from the revised query, the weighted average and a whitening transform in the feature space are found to be the optimal solutions. However, this algorithm does not use the negative examples to update the query and image similarity measure; and initially the user needs to input the critical data of training vectors and the relevance matrix into the system.

Machine learning is about constructing computer programs that can be improved with experience. Any task that can be improved as a result of experience can be considered as a machine-learning task. In CBIR, relevance feedback improves the retrieval performance, and the *experience* is the feedback examples provided by the user. Therefore, relevance feedback can be considered as a learning problem – the system learns from the examples provided as feedback by a user to refine the retrieval results. The aforementioned query-movement method represented by the Rocchio's formula and re-weighting method are both simple learning methods. However, as users are usually reluctant to provide a large number of feedback examples, i.e., the number of training samples is very small. And the feature dimensions in CBIR systems are usually high. Thus, the fact that how to learn from small train-

ing samples in a very high dimension feature space makes many learning methods, such as decision tree learning and artificial neural networks, unsuitable for CBIR.

There are several key issues in addressing relevance feedback in CBIR as a small sample learning problem. First, how to quickly learn from small sets of feedback samples to improve the retrieval accuracy effectively; second, how to accumulate the knowledge learned from the feedback; and third, how to integrate low-level visual and high-level semantic features in the query. Most of the research in literature has focused on the first issue. To address the first issue, Bayesian learning has been explored in research about effective learning algorithms and it has been shown advantageous compared with other learning methods, e.g., [40]. Active learning methods have been used to actively select samples which maximize the information gain, or minimize entropy/uncertainty in decision-making. These methods enable fast convergence of the retrieval result which in turn increases user satisfaction. Chen et al [44] use Monte carlo sampling to search for the set of samples that will minimize the *expected* number of future iterations. Tong and Chang propose in [49] the use of SVM active learning algorithm to select the sample which maximizes the reduction in the size of the version space in which the class boundary lies. Without knowing apriori the class of a candidate, the best search is to halve the search space each time. In their work, the points near the SVM boundary are used to approximate the most-informative points; and the most-positive images are chosen as the ones farthest from the boundary on the positive side in the feature space.

4.1.2 Learning in RBIR

Relevance feedback (RF) has also been introduced in RBIR systems for a dramatic performance boost as it does for the image retrieval systems using global representations. Next, we discuss some learning algorithms in RBIR, while Section 4.2 focus on presenting the relevance feedback technique for CBsIR.

In [56], the authors introduce several learning algorithms using the adjusted global image representation to RBIR. First, the query point movement technique is considered by assembling all the segmented regions of positive examples together and resizing the regions to emphasize the latest positive examples in order to form

a composite image as the new query. Second, the application of support vector machine (SVM) [49] in relevance feedback for RBIR is discussed. Both the one class SVM as a class distribution estimator and two classes SVM as a classifier are investigated. Third, a region re-weighting algorithm is proposed corresponding to the feature re-weighting ones. It assumes that important regions should appear more times in the positive images and fewer times in all the images of the database. For each region, measures of region frequency *RF* and inverse image frequency *IIF* (analogous to the *TF* and *IDF* in text retrieval [33]) are introduced for the region importance. Thus the region importance is defined as its region frequency RF weighted by the inverse image frequency IIF, and normalized over all regions in an image. Also, the feedback judgement is memorized for future use by calculating the cumulate region importance. However, this algorithm only consider positive examples while ignoring the effect of the negative examples in each iteration of the retrieval results. Experimental results on a general-purpose image database demonstrate the effectiveness of those proposed learning methods in RBIR.

4.2 Relevance Feedback for CBsIR

Relevance feedback as an interactive learning technique has been demonstrated to boost performance in CBIR systems [42][52]. Despite the great potential of RF shown in CBIR systems using global representations and in RBIR systems, to the best of our knowledge there is *no* research that uses it within CBsIR systems. Here, we present our solution to improve the retrieval performance of the CBsIR framework discussed in Chapter 3 by using relevance feedback to learn the user's intention. Our relevance feedback approach has three main components: (1) a tile re-weighting scheme that assigns penalties to each tile of database images and updates those tile penalties for all relevant images retrieved at each iteration using both the relevant (positive) and irrelevant (negative) images identified by the user; (2) a query refinement strategy that is based on the tile re-weighting scheme to approach the most informative query according to the user's intention; (3) an image similarity measure that refines the final ranking of images using the user's feed-

back information. Each of these components is explained in details in the following subsections.

4.2.1 Tile Re-Weighting Scheme

Researches in RBIR [46][56] have proposed region re-weighting schemes for relevance feedback (RF). In this research, we design our tile re-weighting scheme that specializes the technique presented in [46] to accomodate our tile-oriented (not region-oriented) HTM approach for CBsIR. It should be emphasized that instead of considering all the images in the database to compute the parameters for region weight [56] (which is computationally expensive), our tile re-weighting scheme uses only the positive and negative examples identified by the user to update the *tile penalty* of the positive images only, which is much more efficient. Moreover, the region re-weighting scheme in [46] uses a predefined similarity threshold to determine whether the region and the image is similar or not, otherwise the comparison of region pairs would become too expensive since images might consist of different and large number of regions. This threshold is sensitive and subject to change for different kinds of image datasets. Thus, how to obtain the right threshold is yet another challenge for the RF method in RBIR. However, our RF method for the CBsIR problem does not need any threshold because the number of obtained tiles is the same and small for each database image and there exists implicit relationship between the tiles, which makes it easier to compare them.

In our system, the user provides feedback information by identifying positive and negative examples from the retrieved images. The basic assumption is that important tiles should appear more often in positive images than unimportant tiles, e.g., "background tiles" should yield to "theme tiles" in positive images. On the other hand, important tiles should appear less often in negative images than unimportant tiles. Following the principle of "more similar means better matched thus less penalty", we assign a *penalty* to every tile that represents the database image for the matching process. User's feedback information is used to estimate the "tile penalties" for all positive images, which also refines the final ranking of images. Note that during the RF iterations, the user does not need to specify which tile of a

certain positive image is similar to the query, which would only make the problem simpler at an additional cost to the user. (Nonetheless, we plan to address this in the future.)

Next, we introduce some definitions used to illustrate the *tile penalty* and formalize the overall RF process.

Definition 1: The distance between two tiles T_a and T_b from images I_a and I_b respectively, is:

$$DT(T_a, T_b) = \frac{\sum_{i=1}^{m} Dist(Feature(T_{a_i}), Feature(T_{b_i}))}{m}$$

where T_{a_i} and T_{b_i} are sub-tiles of T_a and T_b respectively, m is the number of unique leaf nodes in the tiles' tree structures at any hierarchical levels (if already at the leaf level, $m = 1$), the $Dist$ function is to be instantiated with some particular distance measure based on the result of the feature extraction done by the $Feature$ function on the tiles, e.g., BIC's *dLog()* function defined in the previous section. ♠

Definition 2: The penalty for a certain tile i from a database image after k iterations is defined as: $TP_i(k), i = 0, \cdots, NT$, where $NT + 1$ is the number of tiles per database image, and $TP_i(0)$ is initialized as $\frac{1}{NT+1}$. ♠

For instance, in Figure 3.1, $NT + 1 = 1 + 9 + 16$, i.e., is equal to the number of nodes in the tree structure representing the hierarchical partition of a database image; for the lowest level, only unique nodes count.

Definition 3: For each tile from a positive image, we define a measure of the distance DTS between tile T and an image set $IS = \{I_1, I_2, \cdots, I_n\}$. This reflects the extent to which the tile is consistent with *other* positive images in the feature space. Intuitively, the smaller this value, the more important this tile is in representing the user's intention.

$$DTS(T, IS) = \begin{cases} \sum_{i=1}^{n} exp(DT(T, I_i^0)), & \text{if } T \text{ is at full tree level} \\ \sum_{i=1}^{n} exp(min_{j=1..NT} DT(T, I_i^j)), & \text{if } T \text{ is at the subtree level} \end{cases}$$

where NT in this case is the number of tiles at the current subtree level. ♠

Assuming that I is one of the identified positive example images, we can compute the tile penalty of image I which consists of tiles $\{T_0, T_1, \cdots, T_{NT}\}$. The user provides positive and negative example images during each k^{th} iteration of

feedback, denoted respectively as $IS^+(k) = \{I_1^+(k), \cdots, I_p^+(k)\}$ and $IS^-(k) = \{I_1^-(k), \cdots, I_q^-(k)\}$, where $p + q$ is typically much smaller than the size of the database.

Based on the above preparations, we now come to the definition of tile penalty.

Definition 4: For all images (only being positive), the tile penalty of T_i after k iterations of RF is computed (and normalized) as:

$$TP_i(k) = \frac{W_i \times DTS(T_i, IS^+(k))}{\sum_{j=0}^{NT}(W_j \times DTS(T_j, IS^+(k))}$$

where $W_i = 1 - \frac{DTS(T_i, IS^-(k))}{\sum_{j=0}^{NT} DTS(T_j, IS^-(k))}$, acts as a penalty, reflecting the influence of the negative examples. ♠

This implies the intuition that a tile from a positive example image should be penalized if it is similar to negative examples. Basically, we compute the distances DTS between a particular tile T and the positive image set IS^+ as well as the negative image set IS^- respectively to update the penalty of that tile from a positive example image. The inverse of the tile's distance from the negative image set is used to weight its corresponding distance from the positive image set.

Let us now illustrate the above methodology with a simple example, which also motivates the notion of tile penalty. For simplicity, assume that the color palette consists of only three colors: black, gray and white. Figure 4.1 shows the top 3 retrieved images and the user's feedback judgement. Image I_1 is marked as a positive example since it actually contains the query image, which exactly represents the sub-image retrieval problem we are dealing with. Image I_2 is also marked as a positive example because it is the enlargement of the query image (and therefore containing it as well).

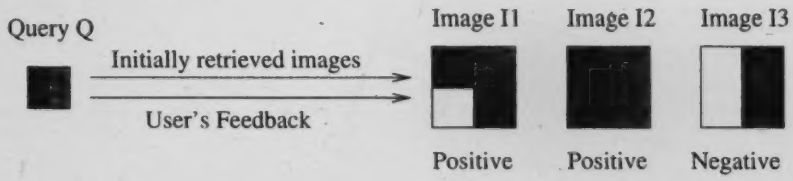


Figure 4.1: Initial set of retrieved images with user's feedback.

For the sake of illustration, assume a two-level multi-cale representation of

database images is used as in Figure 4.2. The tile penalties for tiles per database image are initialized as 0.1 for the 10 tiles, i.e., $TP_i(0) = 0.1, i \in [0, 9]$. Now, take tile T_1 for example. According to Definition 3, we need to compute the distances DTS between T_1 and the positive/negative image set. In order to do this, firstly, the distances between T_1 and all tiles at the corresponding subtree levels of all the images in the positive/negative image set should be obtained by Definition 1. Then, using Definition 4 the new penalty of T_1 is updated from 0.1 to 0.090 correspondingly. The penalties for other tiles is updated in the same way during each feedback iteration. We illustrate the new values of all tile penalties for database image I_1 as a positive example after one feedback iteration in Figure 4.2. We can see that after the user provides feedback information, some tiles lose some weight while others gain. For instance, T_1, T_2, T_3 and T_9 receive less penalties now because they only contain the color of grey and/or black which is/are also in the query. T_0, T_4, T_5, T_7 and T_8 are penalized more since they all contain the color white. The new weights for these tiles generally follow the trend that more percentage of white color more penalty. T_6, which is a rotation of the query image maintains its weight for this iteration. This means that our system is to some extent also capable of perceiving changes such as rotation. Besides, for a closer look at the updated tile penalties of positive image I_1, T_1 receives more penalty than T_3 now although they are similar to the query image in the same degree. Note that, according to Definition 4, both the positive and the negative example images are used to calculate new tile penalties. And we penalize a tile more if it is also somewhat more similar to the negative example images compared with other tiles in the positive example image. Thus it is reasonable that the tile penalty for T_1 appears higher than that for T_3 after feedback learning, since T_1 contains some black color which is also in the negative example image I_3 while T_3 contains only the grey color.

4.2.2 Query Feature Update

The RF process using query refinement strategy is based on the tile re-weighting scheme and all positive and negative example images. The main concern is that we need to maintain as much as possible the original feature of query image while

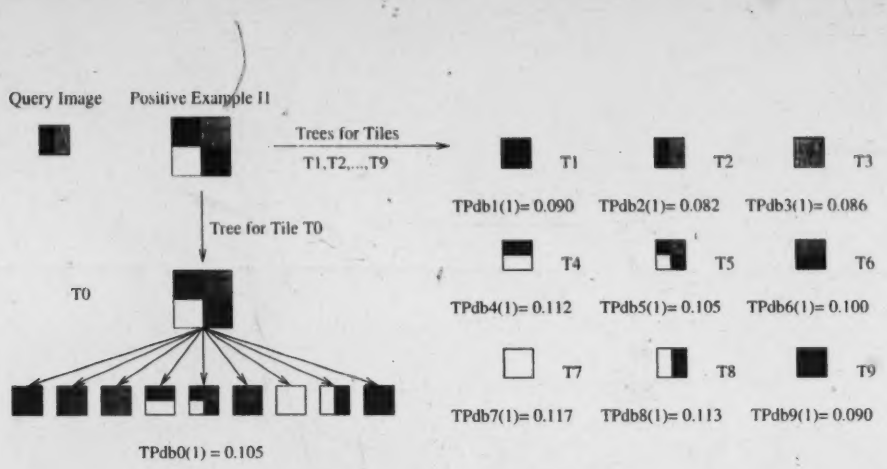


Figure 4.2: Comparison of tile penalty for database image I_1 before and after feedback.

introducing new feature elements that would capture more new relevant images. Considering the hierarchical tree structure of the query image, we use the most similar tile (with minimum tile penalty) at every subtree level of each positive image to update the query feature at the corresponding subtree level.

Definition 5: The updated query feature after k iterations is:

$$qn_l^k[j] = \frac{\sum_{i=1}^{p}(1 - TPmin_{i_l}(k)) \times Pos_{i_l}^k[j]}{\sum_{i=1}^{p}(1 - TPmin_{i_l}(k))}$$

where qn_l^k is the new feature with M dimensions for a subtree (tile) at the l^{th} level of the tree structure for the query image after k iterations, $TPmin_{i_l}(k)$ is the minimum tile penalty for a subtree (tile) found at the l^{th} level of the tree structure for the i^{th} positive image after k iterations, $Pos_{i_l}^k$ is the feature for the subtree (tile) with minimum tile penalty at the l^{th} level of the i^{th} positive image's tree structure after k iterations, and p is the number of positive images given by the user at this iteration. ♠

Intuitively, we use the weighted average to update the feature for a subtree (tile) of the query, based on the features of those tiles that have minimum tile penalties within respective positive images. In this way, we try to approach the optimal query that carries the most information needed to retrieve as many relevant images to the query as possible.

4.2.3 Image Similarity

With the updated query feature and tile penalties for positive images, we can now define the distance between images and the query for ranking evaluation at each feedback iteration. In order to locate the best match to the query sub-image, our image similarity measure tries to find the minimum from the distances between the database image tiles and the query (recall that both the database image and the query sub-image have been modeled by the tree structure in the same way) at corresponding hierarchical level in the tree structure, weighted by the tile penalty of corresponding database image tiles.

Definition 6: The distance between the (updated) query image Q and a database image I at the k^{th} iteration is:

$$DI_k(I,Q) = min_{i=0..NT} TP_i(k-1) \times DT(I_i, Q_j)$$

where $NT+1$ is the number of all subtrees in the tree structure (tiles) of a database image, and $TP_i(k-1)$ is the tile penalty for the i^{th} tile of image I after $k-1$ iterations. ♠

For the comparison of full tree structures, $i = 0$ and $j = 0$, indicating both the full tree structure of the database image and the query image. For the comparison of subtree structures, $i = 1..N_l$ for each $1 \leq j \leq (L-1)$, where N_l is the number of subtree structures at the l^{th} level of the tree structure and L is the number of levels of the tree structure, mapped from the hierarchical partition. j indicates the subtree structure at a particular level of the query image's tree structure, as a result of shrinking the original query tree structure to make the comparison with the subtree structures of database images comparable.

Thus, the overall RF process for the CBsIR system can be summarized in the following pseudo algorithm:

1. The user submits a query (sub)-image with no concern about whether the query is a tile or similar to any tile of any database image;

2. The system retrieves the initial set of images using a similarity measure,

which consists of database images containing tiles similar to the query sub-image;

3. The system collects positive and negative feedback examples identified by the user;
4. For each positive image, update the tile penalties of those tiles representing this image using positive examples and negative examples;
5. Update the query using positive images and their newly updated tile penalties;
6. Use the revised query and new tile penalties for database images to compute the ranking score for each image and sort the results;
7. Show the new retrieval results and go to step 3.

4.3 Performance Measures

Up to now, we have integrated relevance feedback with our CBsIR system via the hierarchical tree matching scheme. Two types of effectiveness for the system should be taken into account. The first one (similar to what was done in the previous chapter) is about retrieving the original images from which the queries are extracted. This is evaluated by using the *average r-measure* and the *average precision* as discussed in Section 3.7. The second type of effectiveness is about retrieving all images relevant to the queries, where it becomes appropriate to calculate the precision and recall for each feedback iteration. For certain applications, it is more useful that the system brings new relevant images (found because of the update of query feature from previous feedback) forward into the top range rather than keeping those already retrieved relevant images again in the current iteration. For other applications, however, the opposite situation applies and the user is more interested in obtaining more relevant images during each iteration including those s/he has already seen before. Besides, it is more helpful that the system learn the user's intention within as fewer iterations as possible. Given these observations, we use two complementary measures for precision and recall as follows:

1. *New Recall*: the percentage of relevant images that were not in the set of the relevant images retrieved during previous iterations over the number of relevant images in the answer set. (Measured only after the first iteration, i.e., after the first feedback cycle.)

2. *New Precision*: the percentage of relevant images that were not in the set of the relevant images retrieved during previous iterations over the number of retrieved images at each iteration. (Also measured after the first iteration.)

3. *Actual Recall*: the percentage of relevant images at each iteration over the number of relevant images in the answer set.

4. *Actual Precision*: the percentage of relevant images at each iteration over the number of retrieved images at each iteration.

The new recall and precision explicitly measure the learning aptitude of the system; ideally it retrieves more new relevant images as soon as possible.

Moreover, we also try to measure the total number of distinct relevant images the system can find during all the feedback iterations. This is a history-based measure that implicitly includes some relevant images "lost" (out of the top presented images) in the process. We call them *cumulative recall* and *cumulative precision* defined as follows:

1. *Cumulative Recall*: the percentage of distinct relevant images from all iterations so far (not necessarily shown at the current iteration) over the number of relevant images in the predefined answer set.

2. *Cumulative Precision*: the percentage of distinct relevant images from all iterations so far over the number of retrieved images at each iteration.

Table 4.1 exemplifies the measures mentioned above, assuming the answer set for a query contains 3 images A, B, C and the number of returned (presented) images is 5.

In addition, we also measure each method's storage overhead and query processing (time) cost.

Table 4.1: Cumulative/New/Actual Recall and Precision

Iteration	Retrieved Relevant Ones	Cumulative Recall/Precision	New Recall/Precision	Actual Recall/Precision
1	A	33.33%/20%	–/–	33.33%/20%
2	A	33.33%/20%	0%/0%	33.33%/20%
3	B,C	100%/60%	66.67%/40%	66.67%/40%

4.4 Experiments and Results

We test the proposed relevance feedback approach for the CBsIR system using the image database mentioned in Section 3.7.2. The broad-domain image dataset consists of 10,150 color JPEG images: a mixture of the public Stanford10k[1] dataset and some images from one of COREL's CD-ROMs, each of which falls into a particular category – we use 21 such categories[2]. Some categories do not have rotated or translated images, but others do. On average, each answer set has 11 images, and none of the answer sets has more than 20 images, which is the amount of images we present to the user for feedback during each iteration. We manually crop part of a certain image from each of the above categories to form a query image set of 21 queries (one for each category). Images of the same categories serve as the answer sets for queries (one sample query and its corresponding answer set are shown in Figure 4.3). The size of the query image varies, being on average 18% the size of the database images. The following performance results are collected from the online demo[3] on a computer running Linux 2.4.17 with two Pentium III CPUs and 256MB of main memory.

In our experiments, the maximum number of iterations explored is set to 10 (users will give feedback 9 times by pointing out which images are relevant (positive)/irrelevant (negative) to the query) and we present the top 20 retrieved images at each iteration. Note that in our system the series of feedback iterations between queries is independent, i.e., the information collected from the user is not integrated

[1] http://www-db.stanford.edu/~wangz/image.vary.jpg.tar.
[2] The union of http://db.cs.ualberta.ca/mn/CBIRone/ and http://db.cs.ualberta.ca/mn/CBIRtwo/
[3] http://db.cs.ualberta.ca/mn/CBsIR.html

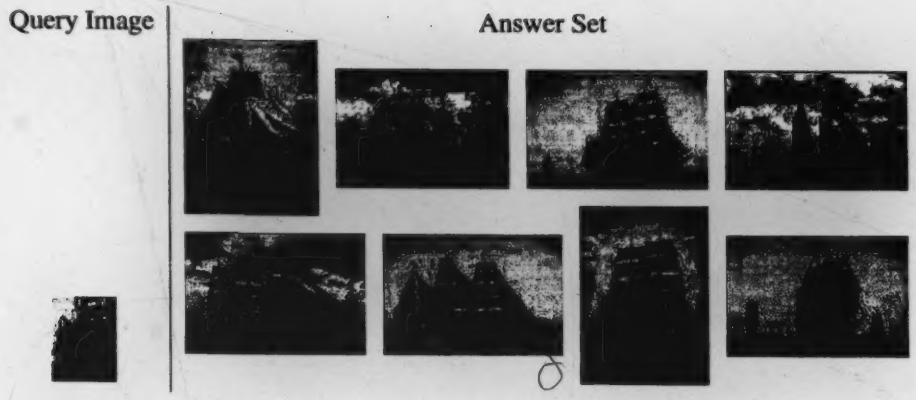


Figure 4.3: A sample query (sub)image and its relevant answer set.

Table 4.2: Comparison of retrieving the original images using BIC by feedback iterations.

BIC parameter (SIZE)	Average No. of Iterations needed for Rank $\leq$ top 20
64 quantized colors	1.1
16 quantized colors	>2.3

into the search for the next queries, even if the very same query is submitted to the system again. This consideration is based on the observation of the subjectivity of human perception and the fact that even the same person would perceive the same retrieval result differently at different times.

Although we have already shown the good retrieval accuracy of finding the original images by using the BIC method for image indexing in our CBsIR system, here we further study the effectiveness of our CBsIR system using BIC to retrieve the original images in terms of feedback iterations. Besides, we tune the BIC parameter (SIZE – the number of quantized colors) to further investigate the HTM-based CBsIR system using the BIC method and relevance feedback technique, comparing the effectiveness as well as efficiency and storage cost for image feature extraction having different degrees of information.

In Table 4.2, it is clear that using the 64 quantized colors the hit rate of the original images can almost reach the optimal value.

For the retrieval accuracy of relevant images using 64 quantized colors in the BIC method, the results are shown in Figure 4.4 and Figure 4.5 by the measures

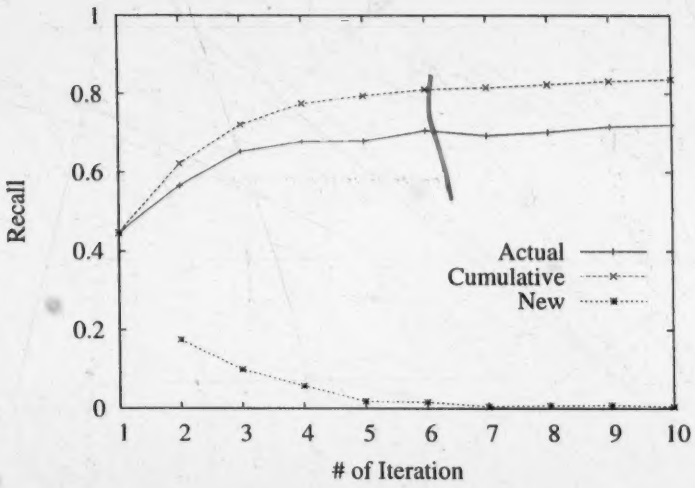


Figure 4.4: Effectiveness measures by actual recall, cumulative recall and new recall using 64 quantized colors in BIC.

proposed in Section 4.3.

As it can be clearly seen that after 5 iterations the system has already learned most of the information it could learn, i.e., the information gain (given by the new recall and new precision curves) is nearly null. On the other hand, after only 5 iterations the actual recall and actual precision values increased by 55% and 60% respectively. It is also noteworthy to mention that the stable actual precision value of nearly 40% is not as low as it may seem at first. The answer sets have an average of 11 images and since the user is presented with 20 images, the maximum precision one could get (on average) would be about 50%. Hence, in this perspective 40% of actual precision is not a low value. Similarly 70% of actual recall means that on average 8 images out of the 20 presented are actually relevant after 5 iterations, which also seems to be quite reasonable. We also obtained about 85% for cumulative recall and about 50% for cumulative precision. The reason for the higher values than those for actual recall and actual precision is because some relevant images that may be "lost" in subsequent iterations are always accounted for in these measures.

Figure 4.6 and Figure 4.7 give the glimpse of screenshots about the online demo using 64 quantized colors in BIC for a sample query during the first two iterations.

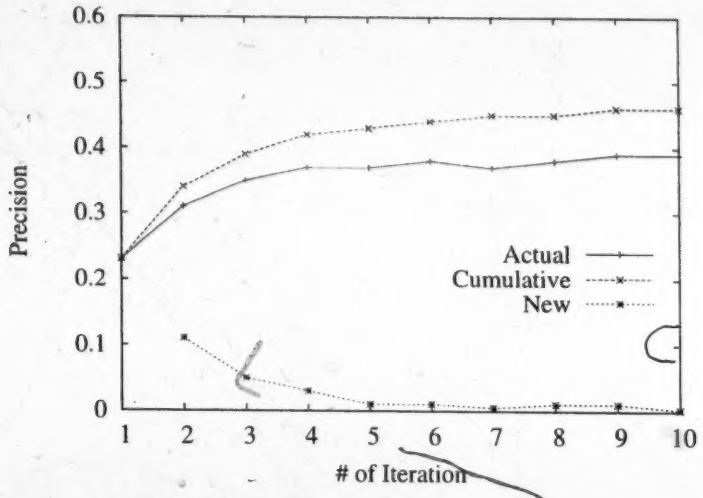


Figure 4.5: Effectiveness measures by actual precision, cumulative precision and new precision using 64 quantized colors in BIC.

On the aspect of effectiveness, using 16 quantized colors expectably yields a worse accuracy than using 64 quantized colors in the BIC method. Looking at Figure 4.8 and Figure 4.9, the former achieves about 60% for the final actual recall and about 30% for the final actual precision, both about 10% lower than the latter as shown in Figure 4.4 and Figure 4.5. The results obtained using 16 quantized colors are also reasonable. The final cumulative recall value increases to 70% from the actual recall value being about 60%. Similarly, the final cumulative precision is brought up to about 40%. The learning aptitudes of the system using different informative degrees of the feature representation (measured by the new recall and new precision shown in Figure 4.8 and Figure 4.9) follow a similar trend, i.e., most of the information is learned by the first 5 iterations. However, it should be noticed that when using 16 quantized colors the left 5 iterations contribute more to reach the final cumulative/actual recall and precision at the 10^{th} iteration, compared with that of using 64 quantized colors. As shown in Figure 4.10 and Figure 4.11, using 16 colors the information gain between the 6^{th} iteration and the 10^{th} iteration is 22.1% for new recall measure and 21.6% for new precision measure; while for 64 quantized colors, these values drop to 10.5% and 15.7% respectively. This is because the result and information gain of using 64 quantized colors are just already

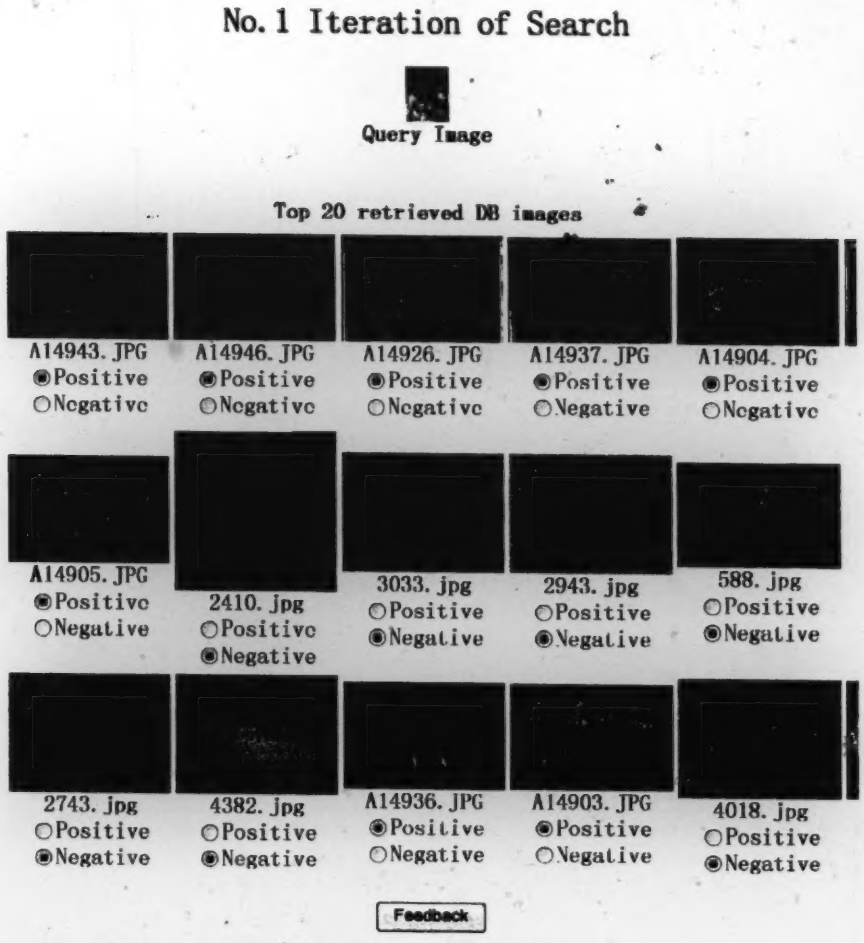


Figure 4.6: Results of online demo using 64 quantized colors in BIC for a sample query after the first iteration. (The user has given feedback once.)

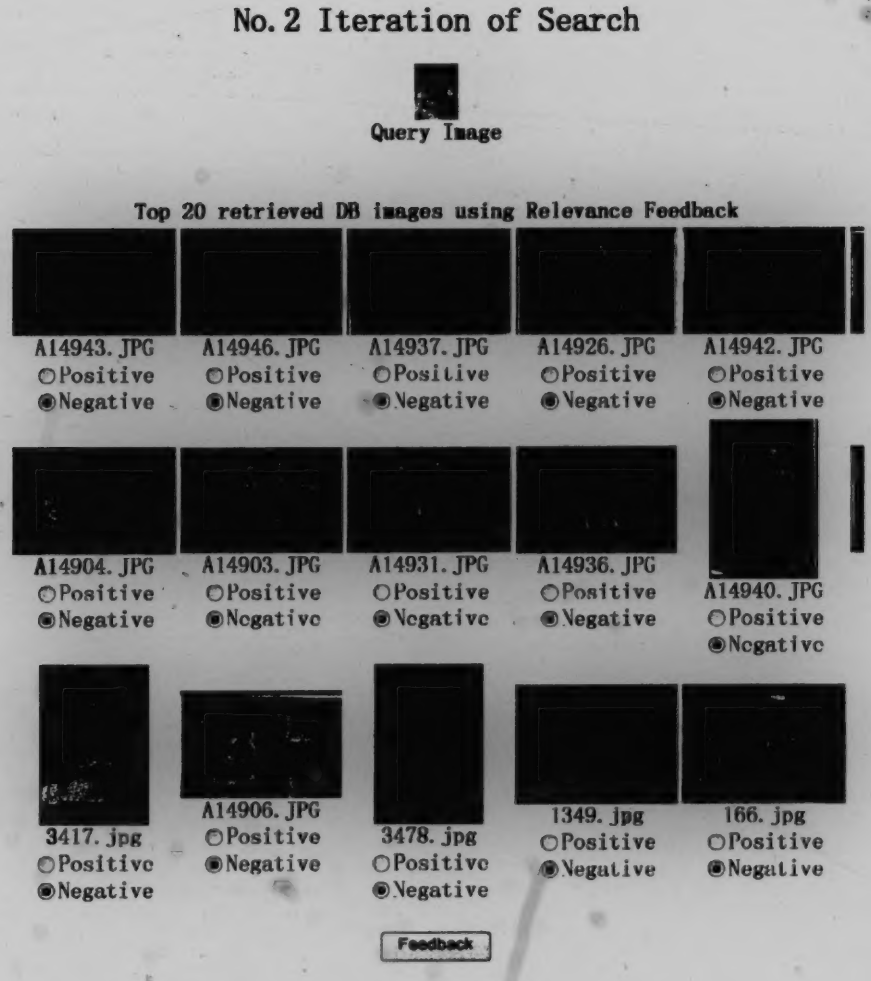


Figure 4.7: Results of online demo using 64 quantized colors in BIC for a sample query after the second iteration.

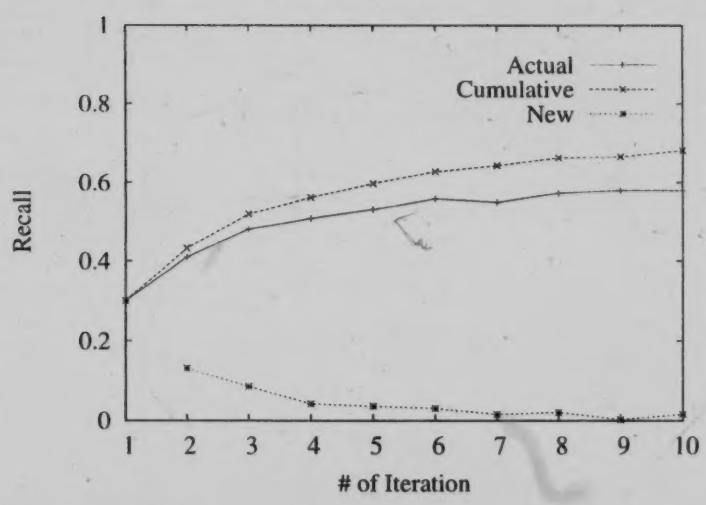


Figure 4.8: Effectiveness measures by actual recall, cumulative recall and new recall using 16 quantized colors in BIC.

better than those of using 16 quantized colors in the previous 5 iterations. Not as much as information left could be learned by the feature representation using 64 quantized colors. The switch point of the two curves is observed in Figure 4.10 and Figure 4.11, appearing in between the 4^{th} and the 5^{th} iterations.

Figure 4.12 shows the average cost, measured in seconds, to process a query during each iteration, i.e., to access all disk-resident data, complete the learning from the user's feedback at the current iteration (not applicable to the first iteration), obtain the distance between the query image and database images and sort them by their resulting ranks. The first iteration takes, on average, slightly less than 2 seconds when using 64 quantized colors and 0.6 second when using 16 quantized colors, whereas each subsequent iteration requires about 2.5 seconds and 1 second respectively for the two feature representations. This slight increase is due to the overhead for computing and updating the tile penalties. If to compare these two image feature representations, using 64 quantized colors is 3.5 times slower than only using 16 quantized colors. With relevance feedback, the difference is narrowed down a little bit with 3 times slower when using 64 quantized colors.

In order to extract image features from the image database applying the BIC method and generate the metadata file, the use of either 64 quantized colors or 16

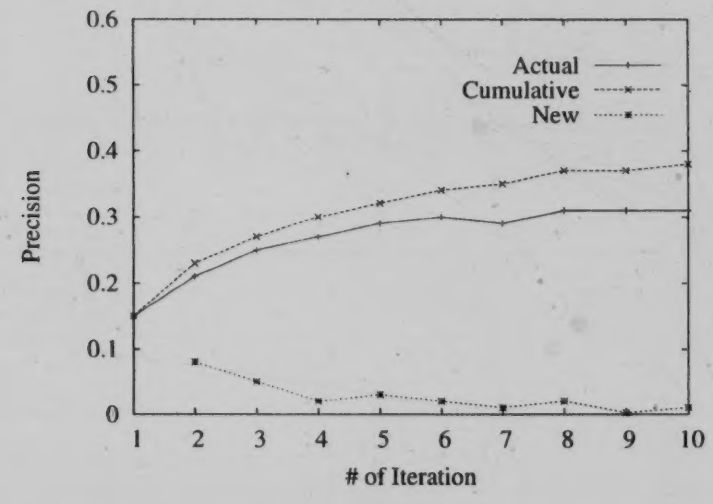


Figure 4.9: Effectiveness measures by actual precision, cumulative precision and new precision using 16 quantized colors in BIC.

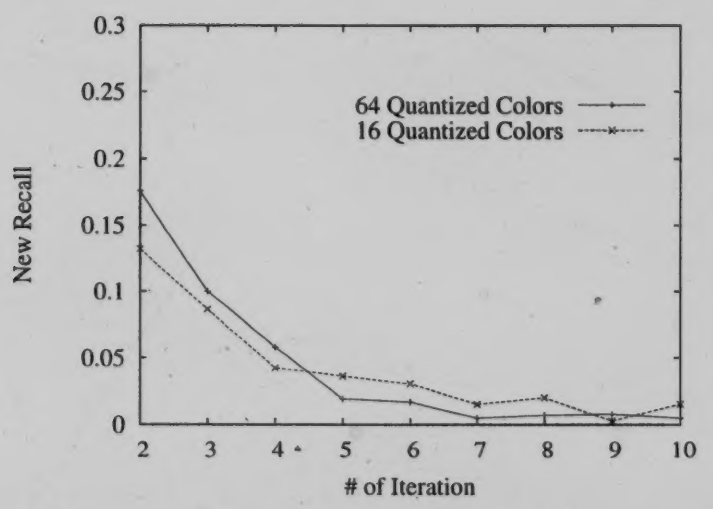


Figure 4.10: New recall (defined from the second iteration) comparison using 64 quantized colors and 16 quantized colors in BIC.

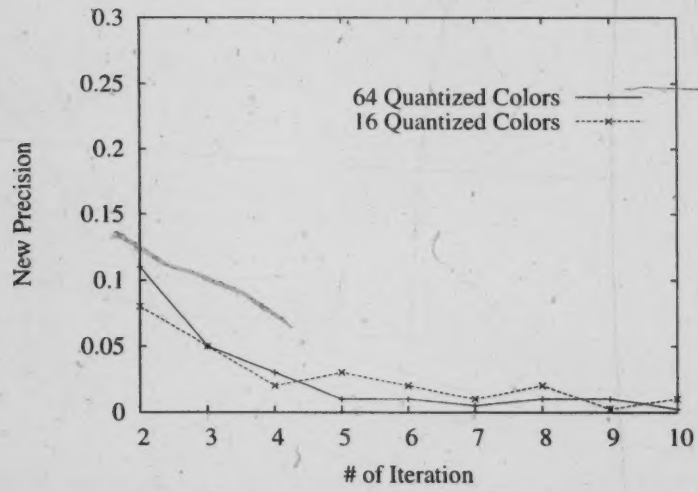


Figure 4.11: New precision (defined from the second iteration) comparison using 64 quantized colors and 16 quantized colors in BIC.

quantized colors requires about 25 minutes on a computer running Linux 2.4.20 with AMD Athlon XP 1900+ CPU and 1GB of main memory. This procedure can nevertheless be done off-line. The storage cost for the disk-resident metadata is 10.5 MB (only about 20% the size of the image database), while using 16 quantized colors needs proportionally less storage, namely 2.7 MB.

In summary, our proposed relevance feedback-based approach for content-based sub-image retrieval (using 64 quantized colors in the BIC method for image indexing) was able to achieve a very good retrieval accuracy with small space cost and fast retrieval time including the overhead due to the feedback learning. When using only 16 quantized colors in BIC, the query processing time cost is cheaper. While the retrieval accuracy suffers, it is still acceptable.

4.5 Summary

In this chapter, we have addressed how relevance feedback can be used to improve the performance of CBsIR. We present the supervised learning method known as *relevance feedback*, which is based on a *tile re-weighting* scheme that assigns penalties to each tile of database images and updates those of all relevant images using

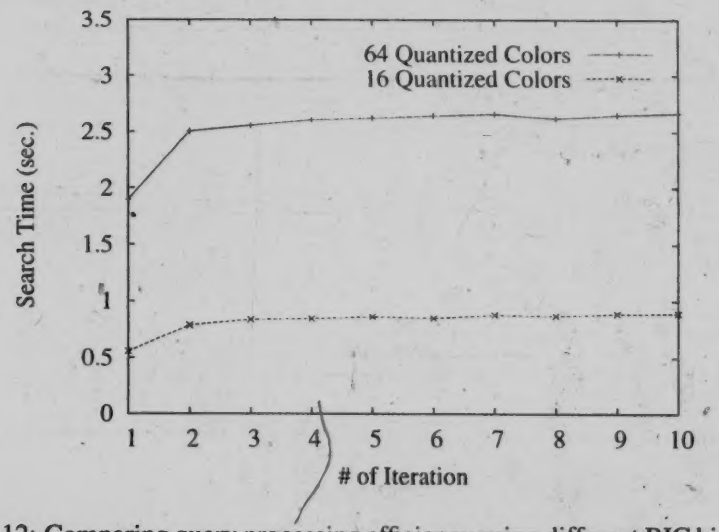


Figure 4.12: Comparing query processing efficiency using different BIC histograms at each iteration.

both the positive and negative examples identified by the user. Moreover, the user's feedback information can also be used to refine the image similarity measure by weighting the tile distances between the query and the database image tiles with their corresponding tile penalties. We combine the learning method with the BIC approach for image indexing to improve the performance of content-based sub-image retrieval. Our results on an image database of over 10,000 images suggest that the learning method is quite effective for CBsIR.

Chapter 5

Conclusions and Future Work

The main contribution of this thesis is the proposal of hierarchical tree matching (HTM) scheme for solving the thus far much less explored problem of content-based sub-image retrieval (CBsIR) and its accompanying issue on object localization. The novelty in this new method is the characterization of images in terms of hierarchical tiles that captures the spatial correlation of the color features and makes for fast and precise object matching. As a summary of the new method, we adopt a multi-scale hierarchical partition to both the database and query images. The average color feature of image tiles is associated with a hierarchical tree structure stored in an index sequence so as to yield fast access during search phase. For the search strategy, we compare the query image's tree structure with the sub-tree structures of the database images at all hierarchical levels and use the average distance between the leaf nodes as the distances between the query tree and the sub-trees of the database images, which introduces local spatial information to provide more veracious matching. Experimental results on a collection of heterogeneous images show that our method achieves both good retrieval accuracy and efficiency. As a side contribution we have shown that one can obtain very different rank measures depending on the distance granularity and ranking criteria in the presence of ties.

Certainly there is room for improvement and a few possible venues for further investigation include the design of disk based access structure for the hierarchical tree (to enhance the scalability for larger databases), the use of better (more powerful yet compact) representation for the tile features, and the incorporation of machine learning techniques to shorten the gap between low-level image features

and high-level semantic contents of images so as to better understand the user's intention. In the latter part of this thesis, we have studies two of the above issues to improve the performance of the CBsIR system using the hierarchical tree matching scheme. On one hand, we adopt the use of a compact and efficient CBIR approach suitable for broad image domains called BIC [54] for image indexing. Experimental results show the BIC feature representation for the image tiles is far more discriminative than the statistic-based feature representation (average color) because of BIC's simple and powerful image analysis algorithm based on a border/interior pixel classification. The logarithmic distance function also helps diminish the distortion in histogram comparison and provide a compact representation of visual features. One other possible improvement of using the BIC approach for feature extraction is to try to solve the problem that image background, which usually covers the majority of image area but does not determine the semantic of the image, could cause distortion in similarity measure by performing "background elimination" during the image analysis process. This action would detect background pixels according to some criterion and exclude these pixels when computing the BIC histogram. In this way, the information that distracts the image semantic is excluded from the image feature representation. Thus, better similarity hit could be expected during the search phase.

On the other hand, the supervised learning method - relevance feedback is investigated to incorporate human perception subjectivity into the retrieval process, trying to capture semantic contents of image in terms of objects. The query refinement method in relevance feedback is integrated with the CBsIR system by applying a tile re-weighting scheme to assign penalties to tiles that compose database images so as to better approach the user's intention. The tile penalties of positive images based on both the positive and negative examples identified by users (without explicit sub-image feedback) are used to update the query for an improvement in the retrieval accuracy of the next iteration. Our experimental results on the general-purpose image database demonstrate the clear performance improvement by this framework compared to that of the previous CBsIR system [57], which uses only average color as the feature representation for image tiles and allowed only one

iteration of retrieval. (Note that as far as we know, there is no research that uses relevance feedback within CBsIR systems and it is not comparable to the region-based retrieval systems although they aim at a similar retrieval goal.) Some venues for future work include integrating other powerful learning algorithms into CBsIR, handling the difference in image resolution between possible queries and target images, and accomplishing a more friendly user interface that allows real time query definition.

In general, the algorithms we propose for the CBsIR problem are not only simple and inexpensive but also quite effective and might be used to automatically solve the adjunctive object localization problem existing in various applications, such as tracking of objects in a video sequence. It is unreasonable to expect any CBsIR system to be absolutely foolproof. However, the goal is to build relatively better CBsIR systems that can offer applications not considered and/or hard to be solved by RBIR with a similar perception on the image content. In this direction, based on the experiments, we feel that there is a compelling reason to use our HTM scheme and corresponding RF approach as one of the basic components in such systems.

Appendix A

Simulation of HTM's Search Process

A high-level simulation of the search process can be viewed in the following series of figures from Figure A.1 to Figure A.5.

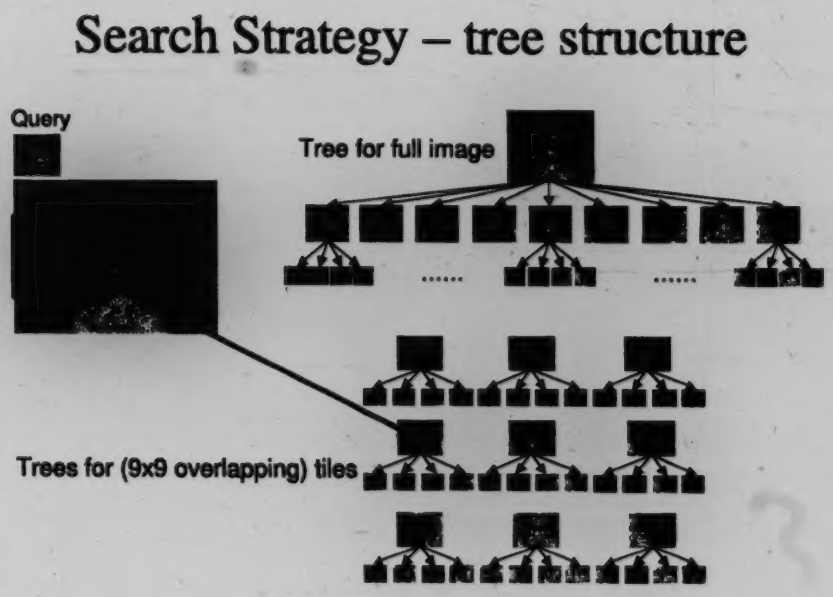


Figure A.1: Tree structures modeling the hierarchical partition of the image.

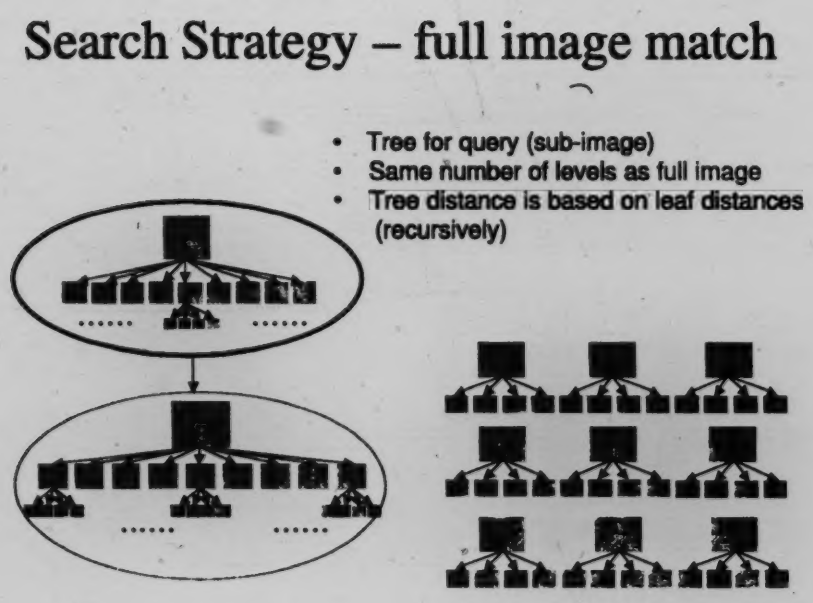

Search Strategy – full image match
• Tree for query (sub-image)
• Same number of levels as full image
• Tree distance is based on leaf distances (recursively)

Figure A.2: Matching the full trees of the query and the database image.

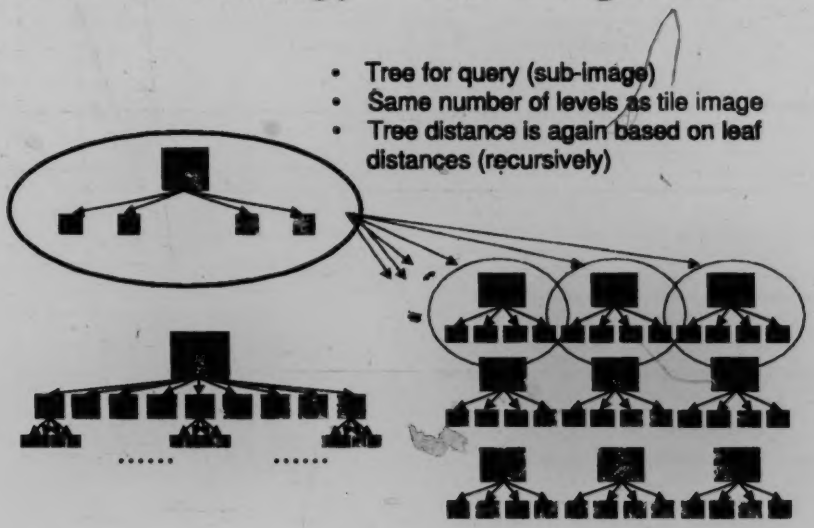

Search Strategy – sub-image match
• Tree for query (sub-image)
• Same number of levels as tile image
• Tree distance is again based on leaf distances (recursively)

Figure A.3: Matching the sub-trees of the query and a certain tile at the second level of the hierarchical partition.

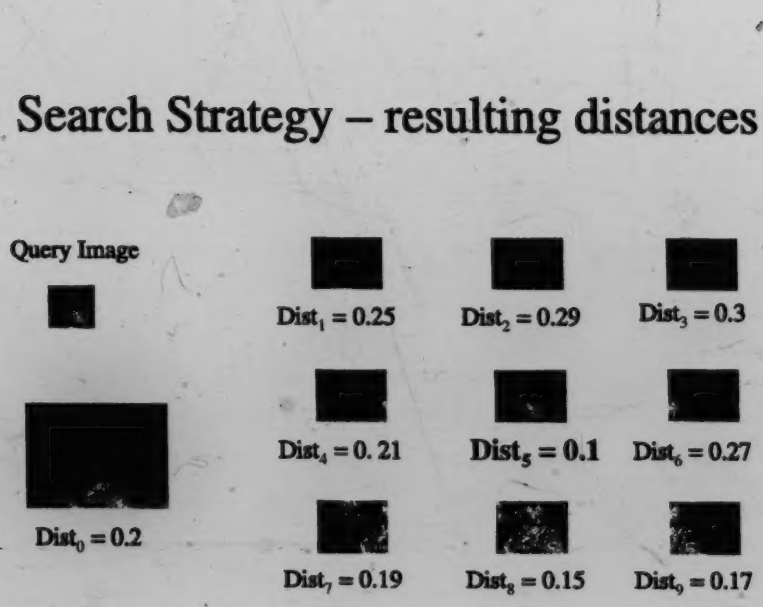


Figure A.4: Obtained distances between the query and each tile of the database image after two kinds of match.

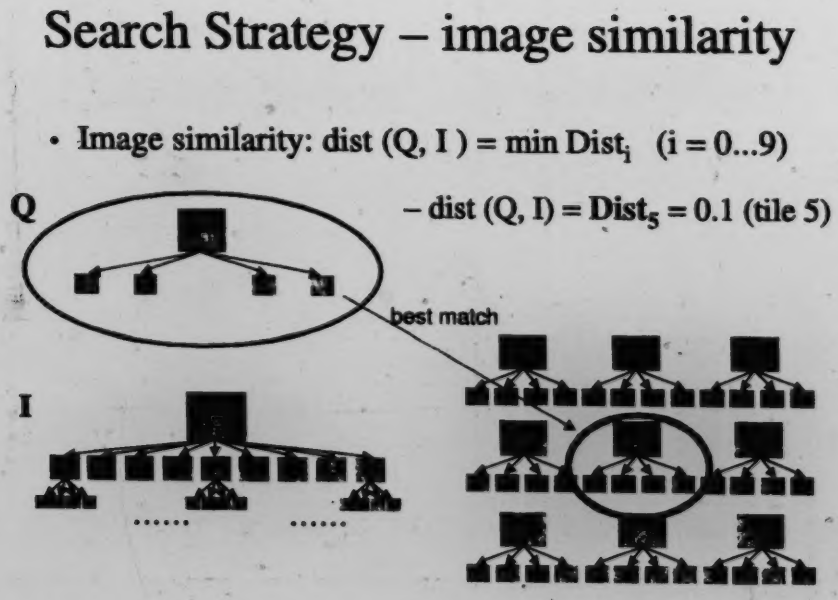


Figure A.5: Image similarity measure - finding the minimum among the obtained distances as the distance between the query and the database image.

Bibliography

[1] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing (Salton, G. eds)*, pages 313–323, Prentice-Hall.

[2] J. C. Falmagne. Psychophysical measurement and theory. In *Handbook of Perception and Human Performance, Vol. I*, chapter 1. Willey Interscience, 1986.

[3] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John Wiley and Sons, 1987.

[4] L. Kaufman and P. J. Rousseuw. *Finding Groups in Data - An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.

[5] H. Samet. em The Design and Analysis of Spatial Data Structures. Addison-wesley Publishing Company, Inc, 1990.

[6] M. Swain and D. Ballard. *Color indexing*. Intl. Journal of Computer Vision, vol. 7, no. 1, pages 11–32, 1991.

[7] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[8] M. Stricker and M. Swain. The capacity of color histogram indexing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 704–708, June 1994.

[9] J. Ashley, R. Barber, M. Flickner, et al. Automatic and semiautomatic methods for image annotation and retrieval in QBIC. In *Proc. of SPIE - Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 24–35, 1995.

[10] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7):729–736, July 1995.

[11] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. The Virage image search engine: An open framework for image management. in *Proc. of SPIE Storage and Retrieval for Image and Video Databases IV*, San Jose CA, USA, pages 76–87, 1996.

[12] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In Proc. of ACM Multimedia 96, pages 65–73, 1996.

[13] J. -Y. Chen, C. A. Bouman, and J. P. Allebach. Multiscale branch and bound image database search. In *Storage and Retrieval for Image and Video Database V, Proc. of SPIE 3022*, pages 133-144, 1997.

[14] W. Grosky. Managing multimedia information in database systems. *Communications of the ACM 40*, No. 12, December 1997.

[15] W. Grosky, R. Jain, and R. Mehrotra. *The handbook of multimedia information management*. Prentice-Hall, Inc. 1997.

[16] J. Huang, S. R. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proc. of the IEEE Comp. Soc. Conf. Comp. Vis. and Patt. Rec.*, pages 762–768, 1997.

[17] W. Ma. NETRA: a toolbox for navigating large image databases. Ph.D dissertation, Dept. of Electronical and Computer Engineering, Univ. of California at Santa Barbara, 1997.

[18] Y. Rui, T. S. Huang, and S. Mehrotra. Content-base image retrieval with relevance feedback in MARS. In *Proc. of the IEEE Intl. Conf. on Image Procssing*, pages 815–818, 1997.

[19] S. Smith and J. Brady. Susan - a new approach to low level image processing. In *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[20] M. Stricker and A. Dimai. Spectral covariance and fuzzy regions for image indexing. *Machine Vision and Applications*, 10:66-73, 1997.

[21] S. Sclaroff, L. Taycher, and M. L. Cascia. ImageRover: a content-based image browser for the World Wide Web. Boston University, CS Dept. *Technical Report* 97-005, 1997.

[22] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using EM and its application to content-based image retrieval. In *Proc. of the 8th Intl. Conference on Computer Vision*, pages 675–682, 1998.

[23] Y. Gong. *Intelligent image databases: towards advanced image retrieval*. Kluwer Academic Publishers, 1998.

[24] V. Gaede and O. Günther. Multidimensional access methods. ACM Computing Surveys, 30(2):170–231, 1998.

[25] J. Huang. Color-spatial image indexing and applications. Ph.D dissertation, Dept. of Computer Science, Cornell University, 1998.

[26] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Query Database Through Multiple Examples. In *Proc. of the 24th Intl. Conf. Very Large Database (VLDB)*, pages 218–227, 1998.

[27] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Condensing image databases when retrieval is based on non-metric distances. In *Proc. of the Intl. Conf. on Computer Vision (ICCV)*, pages 596–601, 1998.

[28] K-S. Leung and R. Ng. Multiresolution subimage similarity matching for large image databases. In *Proc. of SPIE - Storage and Retrieval for Image and Video Databases VI*, pages 259–270, 1998.

[29] P. Montesinos, V. Gouet, and R. Deriche. Differential invariants for color images. In *Proc. of the 14th Intl. Conf. on Pattern Recognition*, 1998.

[30] W. Niblack, X. Zhu, J. L. Hafner, et al. Updates to the QBIC system. In *Proc. of SPIE - Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 150–161, 1998.

[31] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proc. of the 6th Intl. Conf. on Computer Vision*, pages 59–66, 1998.

[32] A. Del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann, 1999.

[33] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addision Wesley, 1999.

[34] C. Carson et al. Blobworld: A system for region-based image indexing and retrieval. In *Proc. of the 3rd Intl. Conf. on Visual Information Systems*, pages 509–516, 1999.

[35] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 1999.

[36] G. Lu. *Multimedia Database Management Systems*. Artech House, 1999.

[37] Y. Rui, and T. S. Huang. A novel relevance feedback technique in image retrieval. In *Proc. of the 7th ACM Conf. on Multimedia*, 67–70, 1999.

[38] J. R. Smith, V. Castelli, and C. S. Li. Adaptive storage and retrieval for large compressed images. In *Proc. of SPIE - Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 467–478, 1999.

[39] N. Sebe, M. S. Lew, and D. P. Huijsmans. Multi-scale sub-image search. In *Proc. of the 7th ACM Intl. Conf. on Multimedia (Part II)*, pages 79–82, 1999.

[40] N. Vasconcelos and A. lippman. Learning from user feedback in image retrieval systems. In *Proc. of NIPS'99*, Denver, Colorado, 1999.

[41] J. Li, J. Z. Wang, and G. Wiederhold. IRM: Integrated region matching for image retrieval. In *Proc. of the 8th ACM Intl. Conf. on Multimedia*, pages 147–156, 2000.

[42] Y. Rui and T. S. Huang. Optimizing learning in image retrieval. In *Proc. of the IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 236–245, 2000.

[43] S. Chan, P. Lewis, K. Martinez, J. Stevenson, and C. Lahanier. Handling sub-image queries in content-based retrieval of high resolution art images. In *International Cultural Heritage Informatics Meeting: Short Paper Track 6*, September 2001.

[44] Z. Chen, W. Liu, F. Zhang, M. Li, and H. Zhang. Web mining for web image retrieval. *Journal of the American Society for Information Science and Technology*, Vol. 52, No. 10, pages 831–839, 2001.

[45] V. Gouet and N. Boujemaa. Object-based queries using color points of interest. In *Proc. of the IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'01*, pages 30–36, 2001.

[46] F. Jing, B. Zhang, F. Lin, W. Ma, and H. Zhang. A novel region-based image retrieval method using relevance feedback. In *Proc. of the 3rd Intl. Workshop on Multimedia Information Retrieval*, pages 28–31, 2001.

[47] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. An adaptive and efficient clustering-based approach for content-based image retrieval in image databases. In *Proc. of the 2001 Intl. Database Engineering and Application Symposium*, pages 356–365, 2001.

[48] R. F. Santos, A. Traina, C. Traina, and C. Faloutsos. Similarity search without tears: The omni-family of all-purpose acceess methods. In *Proc. of the 17th IEEE Intl. Conf. on Data Engineering*, pages 623–630, 2001.

[49] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In Proc. of the 9th ACM Intl. Conf. on Multimedia, pages 107–118, Ottawa, Canada, 2001.

[50] M. A. Nascimento and V. Chitkara. Color-base image retrieval using binary signatures. In *Proc. of the 2002 ACM Symposium on Applied Computing*, pages 687–692, 2002.

[51] M. A. Nascimento, J. Luo, and M. Xue. Scale-preserving sub-image retrieval. *Technical Report*, Dept. of Computing Science, Univ. of Alberta, Canada, 2002.

[52] T. S. Huang, et al. Learning in content-based image retrieval. In *Proc. of the 2nd Intl. Conf. on Development and Learning*, pages 155–162, 2002.

[53] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. Techniques for color-based image retrieval. In C. Djeraba, editor, *Multimedia Mining - A Highway to Intelligent Multimedia Documents*, chapter 6. Kluwer Academics, 2002.

[54] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *Proc. of the 11th Intl. Conf. on Information and Knowledge Management*, pages 102-109, 2002.

[55] T. Wang, J. Shi, and M. A. Nascimento. Experimental results towards content-based sub-image retrieval. In *Proc. of the 2002 IEEE Intl. Conf. on Information Technology: Coding and Computing (ITCC)*, pages 230–235, 2002.

[56] F. Jing, M. Li, L. Zhang, H. Zhang, and B. Zhang. Learning in region-based image retrieval. In *Proc. of the 2nd Intl. Conf. on Image and Video Retrieval*, pages 206–215, 2003.

[57] J. Luo and M. A. Nascimento. Content-based sub-image retrieval via hierarchical tree matching. In *Proc. of the 1st ACM Intl. Workshop on Multimedia Databases*, pages 63–69, 2003.

[58] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. Cell histograms versus color histograms for image representation and retrieval. *Knowledge and Informaiton Systems Journal (KAIS)*, 5(3), pages 315–336, 2003.